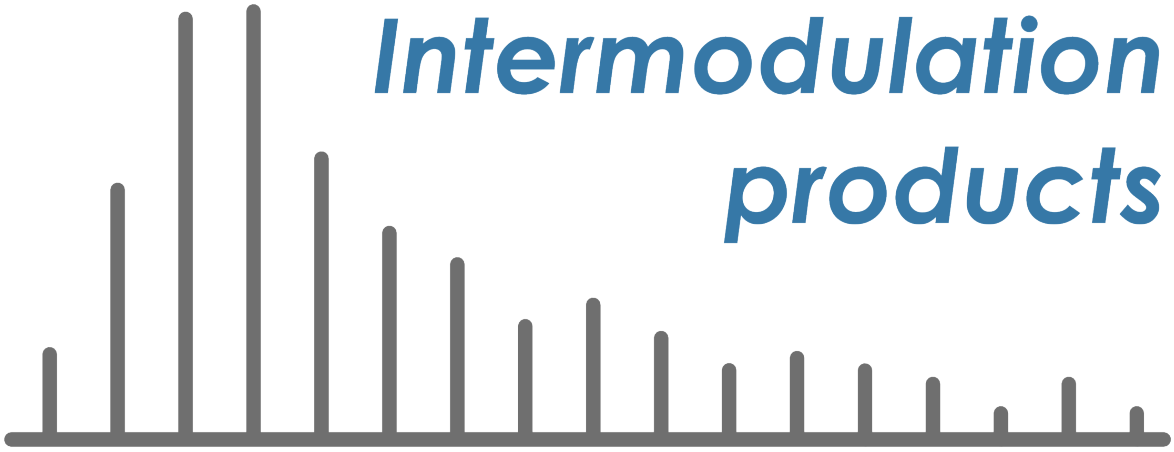


IMP Software Suite User Manual

Release 4.0.2

Intermodulation Products AB

Jan 09, 2020

CONTENTS

1 Introduction **3**
- 1.1 Conventions and useful tips . . . 3
- 1.2 Screen Layout . . . 3
- 1.3 Python . . . 3

2 The IMP Session and Work Flow **5**
- 2.1 Quick Start . . . 5
- 2.2 Work Flow . . . 5
  - 2.2.1 Frequency Sweep . . . 6
  - 2.2.2 Calibration . . . 6
    - 2.2.2.1 Choose Cantilever and Method . . . 7
    - 2.2.2.2 Measure noise . . . 8
    - 2.2.2.3 Fitting noise data . . . 8
    - 2.2.2.4 Calibration Result . . . 8
    - 2.2.2.5 Current calibration . . . 9
    - 2.2.2.6 Add Edit Remove cantilevers . . . 10
    - 2.2.2.7 Calibrating torsional eigenmodes . . . 10
  - 2.2.3 Scanning . . . 10
    - 2.2.3.1 Setup and Scan . . . 11
    - 2.2.3.2 Measure just lifted . . . 11
    - 2.2.3.3 Set-point . . . 12
    - 2.2.3.4 Scan rate . . . 12
    - 2.2.3.5 Image settings . . . 12
      - 2.2.3.5.1 Color Bar . . . 12
      - 2.2.3.5.2 Image Toolbar . . . 13
      - 2.2.3.5.3 Mouse wheel zooming . . . 13
    - 2.2.3.6 Selecting Data in a Scan . . . 14
      - 2.2.3.6.1 Pixel inspector tool . . . 14
      - 2.2.3.6.2 Area inspector tool . . . 14
      - 2.2.3.6.3 Line inspector tool . . . 14
      - 2.2.3.6.4 Clear pixels tool . . . 14
      - 2.2.3.6.5 Level image tool . . . 14
    - 2.2.3.7 Status Banner . . . 14
  - 2.2.4 Analyze Scan Data . . . 15
    - 2.2.4.1 Image Smoothing . . . 15
  - 2.2.5 Session Overview . . . 15
    - 2.2.5.1 Session . . . 16
    - 2.2.5.2 Import host AFM data . . . 16
    - 2.2.5.3 Flip and swap scans . . . 16
    - 2.2.5.4 Batch process parameter maps and force volume data . . . 16
    - 2.2.5.5 Three dimensional viewer . . . 17
  - 2.2.6 Session Logbook . . . 18

3 Quantitative Analysis **19**

3.1 Inspecting single pixels . . . 19
3.1.1 Signal Inspector . . . 19
3.1.2 Force Inspector . . . 20
3.1.2.1 Dynamic force quadratures . . . 20
3.1.2.2 Dissipated work and average potential . . . 20
3.1.2.3 Reconstructed force . . . 20
3.1.2.3.1 Force Reconstruction Methods . . . 21
3.1.2.4 Force Inspector settings . . . 23
3.1.2.5 Background force compensation . . . 23
3.2 Analyzing lines and surfaces . . . 24
3.2.1 Analyzing Linear Transects . . . 24
3.2.2 Parameter Maps . . . 24
3.3 Special Modes . . . 25
3.3.1 Intermodulation Electrostatic Force Microscopy . . . 25
3.3.1.1 Setting up the measurement . . . 26
3.3.1.2 Analyzing and viewing the ImEFM data . . . 27

4 Advanced Topics **29**
4.1 Intermodulation Measurement . . . 29
4.1.1 Time mode . . . 30
4.1.2 Frequency mode . . . 30
4.1.3 Feedback . . . 30
4.2 Noise Calibration . . . 30
4.2.1 Note on sensitivity and accuracy . . . 32
4.3 Programming your own Force Models . . . 33
4.4 Data Tree . . . 33
4.5 Advanced Setup . . . 35
4.5.1 Frequency tuning . . . 35
4.5.2 Setup Feedback . . . 36
4.5.3 Measure . . . 36
4.5.4 Units . . . 37
4.5.5 Manual Setup . . . 37
4.5.6 Out of Range Error . . . 38
4.6 Drive Constructor . . . 38
4.6.1 Configuring the MLA™ . . . 38
4.6.1.1 Pre-defined variables . . . 38
4.6.1.2 Built-in functions . . . 39
4.6.2 Synthesize and Configure . . . 40
4.6.3 Setting the feedback . . . 40
4.6.4 Debugging a script . . . 40
4.6.5 Useful Python code blocks . . . 40
4.7 Stream Recorder . . . 41
4.7.1 Example script . . . 42
4.8 Scripting Interface . . . 43
4.8.1 RecorderAnalysis.py . . . 44
4.8.2 LiveParampeterMap.py . . . 44
4.9 ScanData . . . 44
4.9.1 ScanData Class . . . 44
4.10 File Management . . . 58
4.10.1 File Types . . . 59
4.11 Panels and Views . . . 59

5 Installation **61**
5.1 Install Software . . . 61
5.2 Install Hardware . . . 61
5.2.1 Connection to computer . . . 61
5.2.2 Connection to host AFM . . . 62
5.3 Set AFM Type . . . 62

5.3.1 Working without triggers . . . 63
5.4 Common AFMs . . . 63
5.4.1 Nanoscope ™ III, IIIa, IV . . . 63
5.4.2 Nanoscope ™ V . . . 63
5.4.2.1 software settings . . . 64
5.4.3 Asylum MFP-3D ™ . . . 64
5.4.4 Asylum Cypher ™ . . . 64
5.4.5 JPK Nanowizard ™ III . . . 65
5.4.6 NanoTec Electronica . . . 65
5.4.6.1 Hardware Connection images . . . 65
5.4.6.1.1 SAM III connections . . . 65
5.4.6.1.2 Nanoscope III and IIIa Trigger Access . . . 65
5.4.6.1.3 Nanoscope V Connections . . . 65
5.4.6.1.4 JPK Nanowizard III Connections . . . 65

6 Multifrequency Lockin Amplifier 71
6.1 Version III . . . 71
6.2 Version II . . . 71
6.3 Version I . . . 71
6.3.1 Version I Firmware . . . 72

7 Trouble Shooting 73

8 Changelog 75

9 References 81

Bibliography 83

Index 85

Contents:

CHAPTER

ONE

INTRODUCTION

The **Intermodulation Products AFM Software Suite** (AFM Suite) from Intermodulation Products AB is a collection of software tools for performing **Intermodulation Atomic Force Microscopy** (ImAFM™) – a powerful quantitative surface analysis method. This manual explains how to use the IMP Suite through its Graphical User Interface (GUI). The manual is used most effectively when the IMP Suite is open and running, taking data with your host AFM or analyzing already scanned data.

1.1 Conventions and useful tips

The IMP help icon appears at several places in the AFM Suite. Clicking on this icon will open a browser and jump to the appropriate place in the manual.

`Screen text` is denoted with a shaded box and it should coincide exactly with text in the AFM Suite; on a button, or next to a check box or input field.

The genindex contains key words and important screen text. Each entry is linked to the appropriate place in the manual. If you are wondering about the function of a button in the software use the help icon or search for the button name in the genindex.

Linked text (e.g. *The IMP Session and Work Flow*) cross-references to different parts of the manual. Use the back button on your browser or PDF viewer to return to the jump point.

Bullet lists are used to describe related items which appear in a group. For example:

- Radio buttons
- Control settings
- output fields
- All related to a particular task.

1.2 Screen Layout

The screen layout in the software can be customized to suit the users needs, and the actual appearance depends on the computer platform used (Linux, Windows, Mac). A detailed discussion of how to customize the screen layout is given in the advanced section on *Panels and Views*. For this reason we do not make extensive use of pictures or screen shots in the manual, but rather `screen text` and icons .

1.3 Python

The AFM Software Suite is written in the computer language called Python, an open source language with many powerful tools for plotting, image analysis, and numerical calculation. These are collected in a set of modules called SciPy (Scientific Python)). The AFM Suite is platform-independent, running on Windows, Macintosh or

Linux. Native graphics environments are used in the GUI which is written in wx-Python , so icons and panels may appear slightly different on different platforms.

This manual is written in restructured text language, and is available as HTML or PDF. The HTML version is linked with the help icons in the AFM Suite and it is more easily used when working with the Suite online. A separate software developer documentation describes the usage and function of the Python objects which make up the IMP Suite. The developmer documentation is designed for users who want to enhance the functionality of the software by modifying the source code and programming their own measurement and analysis methods.

CHAPTER

TWO

THE IMP SESSION AND WORK FLOW

2.1 Quick Start

Connect the intermodulation lockin to your host AFM and make sure the switch(s) on your signal access module are in the correct position. For details on connecting to your AFM, please consult the *Install Hardware* section.

Start the host AFM and set it to **Contact Mode**, with the **feedback set-point set to zero volts**. Here we assume that zero volts on the detector corresponds to the equilibrium position of the cantilever when it is free from the surface.

Start AFM Software Suite. A **session folder** named with the current date is automatically created where all your data will be stored.

2.2 Work Flow

The work flow is symbolically represented by the left column of 6 Im icons. From top to bottom, these icons guide the work flow in the AFM Suite. Click on these icons to open the different panels and views.

Find the resonance by performing a *Frequency Sweep*.

Calibrate the cantilever and detector using non-invasive thermal noise *Calibration*.

Setup and scan then engage the surface with the host AFM to start *Scanning*.

Open previous scan files and use the pixel and line inspectors to *Analyze Scan Data*.

View the session and import host AFM data, batch process, or 3D view in *Session Overview*.

Add comments to the automatic logging in the *Session Logbook*.

Each step in the work flow is described in detail in the following sections:

2.2.1 Frequency Sweep

After loading the cantilever and adjusting the detector, find the cantilever resonance by sweeping the frequency of the cantilever drive signal while measuring the detector response signal. The `Frequency Sweep` panel opens when you click the top icon in the work-flow.

`Run sweep` starts the measurement between the `Start frequency [kHz]` and `Stop frequency [kHz]`. Edit these fields to change the sweep range. The sweep range should include the likely resonance frequency of the cantilever, which should be provided by the probe manufacturer and is usually printed on the probe box. Reducing the measurement `Bandwidth [kHz]` causes the sweep to be slower and the measured response to be more accurate. The sweep speed also depends on the `Number of points` recorded during the sweep. The measurement time at each point is given by the inverse of the bandwidth. You may need to adjust the `Drive amplitude [V]` in order to the right amount of excitation to excite the cantilever resonance.

Check boxes provide automatic features for finding the resonance and controlling the sweep:

- `Zoom peak` performs a second sweep with finer detail, zoomed in on the peak response found in the sweep range.
- `Transfer results to noise calibration` transfers the center frequency and estimated sweep range to the next stage in the work flow.
- `Reverse sweep direction` gives a sweep with increasing and then decreasing frequency over the range specified.

2.2.2 Calibration

Quantitative AFM starts with a good calibration of the cantilever and ImAFM™ requires this calibration *before* you start to scan a surface. The AFM Software Suite contains the latest methods for cantilever calibration based on the measurement of the thermal Brownian motion of the cantilever and a theory of hydrodynamic damping of the oscillating beam. By fitting a theoretical model to the noise data we determine all constants necessary to the measure the cantilever deflection in meters, and convert this deflection to force in Newtons.

The IMP Suite determines the parameters of the cantilever as a *dynamic* transducer of force. ImAFM™, being a dynamic method of force measurement, requires the stiffness of the beam in the frequency band of the measurement (near resonance), which is not the same as the static bending stiffness of the beam. The method works with the fundamental eigenmode of the cantilever, or the resonance with lowest frequency. Details and references to the literature are given in the advanced section on *Noise Calibration*. Here we describe how to perform the calibration.

The tab at the very top selects between calibration of the `Flexural` or `Torsional` eigenmode. The latter calibration is available if you have the Intermodulation Frictional Force Microscopy (ImFFM) option. The calibration procedure is very much the same for either type of mode, but the underlying formulas and analysis are different. We begin by describing calibration of the flexural eigenmode, and end with a discussion of the torsional eigenmode.

Note: For the calibration to be accurate, is important that the calibration is performed when the cantilever is well free of the surface, at least several cantilever widths or about 0.1 mm or more from the surface.

2.2.2.1 Choose Cantilever and Method

`Calibration Parameters` specifies the type of cantilever and calibration method.

- `Temp [C]` is the temperature of the damping medium which is in thermal equilibrium with the cantilever. This temperature determines the magnitude of the thermal noise fluctuation force that is driving the cantilever.
- `Fluid` selects the density and viscosity of the damping medium surrounding the beam, needed for calculation of hydrodynamic damping. Noise calibration methods have not been tested thoroughly beyond studies in `Air`, but we include to option to apply the theory in `Water 20 C`.
- `Cantilever` selects from a list of cantilevers for which calibration constants have been published. If your cantilever is not on this list you can choose `Arbitrary Rectangular` and enter the cantilever `Length, L [um]` and `Width, b [um]` of your rectangular beam cantilever. You can edit the length and width fields for any chosen cantilever without changing the stored values for that cantilever. Alternatively, you can *Add Edit Remove cantilevers* as described below.
- `Method` selects between six methods of calibration:
  - `Hydrodynamic function` uses an analytic expression for the hydrodynamic function to calculate the damping. The calculation applies to a rectangular beam in the limit, $L >> b$ (see [Sader-1998]). This method uses length and width given in the fields above.
  - `Sader constants` uses a good approximation to the hydrodynamic function described with three parameters a_0, a_1, a_2 that are specific to the particular type of cantilever. These constants have been measured and checked against other methods and according to theory they should apply to any cantilever of the same shape (see [Sader-2012]). For this method, the length and width fields have no influence.
  - `Reference calibration` brings up three fields. If you have one good calibration for a particular type of cantilever made with some other means, such that you know all three constants: the quality factor, stiffness and resonant frequency, or `Q-ref`, `k-ref` and `f0-ref` respectively, you can use this calibration as a reference for calibrating other cantilevers of the same shape (see [Sader-2012]). For this method, the length and width fields have no influence.
  - `Thermal tune: responsivity` brings up a field to enter the known stiffness of the cantilever fundamental eigenmode, `k [N/m]`. The thermal noise measurement is then applied to determine the detectors inverse responsivity [nm/V].
  - `Thermal tune: stiffness` brings up a field to enter the known detector inverse responsivity, `Inv. resp. [nm/V]`. The thermal noise measurement is then applied to determine the cantilever mode stiffness.

  - `Manual` overrides the thermal noise measurement and no hydrodynamic damping theory is applied. All four calibration constants are entered in the fields given: the eigenmodes resonance frequency `f0 [Hz]`, dimensionless quality factor `Q [-]`, mode stiffness `k [N/m]`, and the detectors inverse responsivity `Inv. resp.[nm/V]`.

2.2.2.2 Measure noise

`Acquire Data` controls the start of measurement, saves data, or loads previous measurement data.

- `Run calibration` starts the averaging of many separate noise measurements, to decrease the fluctuations in the noise data. If *Frequency Sweep* was used to find the resonance and `Transfer results to noise calibration` was activated, the software will automatically analyze an appropriate frequency range around the resonance. The *Status Banner* indicates the progress of the calibration.
- `Settings` opens an `Acquisition Settings` dialog box where you can select the total number of `Measurements` in the average, the `Center frequency [kHz]` and `Frequency span [kHz]` to analyze, and the `Frequency resolution [Hz]` between data points. You can also give a `Down-sampling` factor which averages the given number of samples before transferring to the computer for spectral analysis.
- `Save As` opens a dialog box to save the noise data to a .txt file. If `Autosave` is checked (on the bottom *Status Banner* of the main frame), calibration files will be automatically saved. The file is saved in the JSON format, and it can easily be opened in many different programing languages (Matlab, Python, Java, etc.) It is not necessary to save the raw noise data for each calibration, as the software automatically keeps track of the *Current calibration*, or most recent fit of calibration data, which is stored as the relevant calibration in the scan file.
- `Load` opens a dialog box to load a previously saved noise data.

2.2.2.3 Fitting noise data

When the `Enable Fit` box is checked, the fit is performed in real time and displayed in the plot as a solid blue line. The noise contribution from the cantilever motion is shown with the yellow dash line. For a full description of the theory and fit, see *Noise Calibration*.

Sometimes very low level spurious signals can be picked up when measuring noise. If the spurious signal is not actually exciting the cantilever (i.e. present on the drive voltage), then you can improve the calibration by simply removing it from the fit. Select the following tools in the calibration plot toolbar:

- right-click-and-drag on the plot will select a range where the fit will be performed. Data outside this range becomes gray and is not analyzed in the fit.
- clears the selected fit range.
- right-click-and-drag defines an area over which data points are ignored in the fit. The removed data is marked with a red x.
- clears all ignored data.

2.2.2.4 Calibration Result

The `Calibration Result` is displayed in several ways, including important figures of merit to judge the quality of the calibration. It is useful to think of the calibration as having two parts: The calibration of the cantilever (the force transducer) and the calibration of the detector (the deflection sensor, or optical lever).

`Cantilever:`

- `Resonance frequency` $f_0 = \sqrt{k/m}$ [kHz] of the simple harmonic oscillator model used to model the cantilever eigenmode with mode stiffness k and effective mass m.

- `Width of resonance` γ [Hz] related to the damping coefficient of the simple harmonic oscillator model. $1/\gamma$ is the characteristic time for exponential decay of oscillation amplitude due to the damping of the surrounding medium.
- `Quality factor` $Q = 2\pi f_0/\gamma$ is the ratio of the energy stored in the oscillation, to the energy lost per cycle of oscillation. A freely oscillating cantilever will 'ring' for Q cycles before the amplitude decays by a factor of $1/e$, or 37%.
- `Mode stiffness` k [N/m] is the eigenmode stiffness, or dynamic spring constant, which is the coefficient of the linear restoring force in the simple harmonic oscillator model describing the eigenmode.
- `Thermal noise force` [fN/$\sqrt{\text{Hz}}$] is the fluctuation force associated with the linear, viscous damping of the cantilever. This frequency-independent force noise gives rise to a peak in the deflection noise near a high-Q resonance. The thermal noise force represents a fundamental sensitivity limit for force measurement (see *Note on sensitivity and accuracy*).

`Detector:`

- `Inverse responsivity` $|\alpha|^{-1}[\mu m/\text{V}]$ is the inverse magnitude of the detectors response function, which converts the measured signal in Volts [V] to a deflection of the tip in meters [m]. In the literature, this constant is often referred to as the 'inverse optical lever sensitivity' (invOLS), but we prefer the term 'responsivity' (see *Note on sensitivity and accuracy* for further discussion) . Actually, the calibration internally in the software does not rely on a calibrated measurement of Volts by the MLA™, nor does it rely on the calibration of the AFM scanner to determine detector responsivity. The software uses the noise measurement to determine cantilever deflection in the digital counting units of the MLA™, or Ananog-to-Digital units [ADU]. Thus, the noise measurement and hydrodynamic theory together comprise a more *primary* form of calibration. All force and distance data given by the AFM Software Suite are derived from this calibration, independent of calibration error in the measurement of detector signal in Volts, or movement of the scanner in nm. The thermal noise method does however depend (weakly) on a calibrated measurement of temperature.
- `Noise floor` [fm/$\sqrt{\text{Hz}}$] is the detector noise, expressed as an equivalent deflection noise of the cantilever. The noise floor is a good figure of merit for expressing the sensitivity of the opto-electronic system which detects the cantilever deflection. Note that this noise floor will depend on the type of cantilever (in particular the length) and on the adjustment of the detector (location of laser spot).
- `Equivalent force` [fN/$\sqrt{\text{Hz}}$] expresses the detector noise as and equivalent force noise acting on the cantilever. This equivalent force noise gives the sensitivity of force measurement, if the same cantilever were to measure force quasi-statically, or at low frequency well below resonance.
- `Peak to flat ratio` [dB] is the ratio of the noise peak (thermal noise) to the flat background noise (detector noise). If you compare measurements with the same cantilever and two different detectors, the larger this number, the better the detector (see *Note on sensitivity and accuracy*).

Finally, the `Calibration` panel shows three plots of `Stiffness`, `Res. freq.` and `Q-factor` which are updated as the data is averaged and re-anlayzed. These plots allow you to judge how many measurements are required to converge to a stable value.

2.2.2.5 Current calibration

Running a new calibration, loading a previous calibration, selecting a new range or excluding points, runs the fitting routine which determines all the calibration constants. These constants define the **current calibration**. After each scan, all calibration constants from the current calibration are stored with the scan file, together with the raw spectral data acquired by the MLA™. It is not necessary to store a calibration file in order to analyze the data in scan file. The `Store` and `Load` features are there only if you want to store or view raw noise data for a particular run of the calibration. The raw noise data and the calibration plot will be saved for each run if `Autosave` is checked. It is very easy to re-calibrate during a scan session and thereby re-define the current calibration. Simply stop the scan, retract the probe well away from from the surface (at least several cantilever widths, at least 0.1 mm), and re-run the calibration. We recommend that you re-calibrate frequently, to check if anything has changed with the cantilever and detector. Calibration should also be re-run if any adjustments have been made to the laser or the detector, and it must be performed with every new cantilever, before you start to scan with ImAFM™.

2.2.2.6 Add Edit Remove cantilevers

In the `Calibration Parameters` group, the `Cantilever` selector has the option to `Add / Edit / Remove` cantilevers. This option does not change the selected cantilever. It opens a dialog box that allows you to change calibration parameters of the selected cantilever, or create a new cantilever.

The `Add` tab has two options for creating a new cantilever:

- `Basic` is for rectangular cantilevers. Enter the `Length, L [um]` and `Width, b [um]` in micrometers. Make sure that you enter a `Cantilever name` and the `Manufacturer`. Click `Add` and your cantilever will appear in the pull-down list. The calibration method will will be based on the theoretical expression for the hydrodynamic damping function, valid for $L >> b$ (see `Method` above).
- `Advanced` is for cantilevers of arbitrary plan-view dimensions. Here you can enter the `Sader Constants` if they are known, or the calibration parameters of a `Reference calibration` for a cantilever of the same plan view dimensions (see `Method` above). Length and Width are interpreted as effective values for a non-rectangular cantilever. The `Shape` selector is used to specify that the cantilever is, or is not rectangular.
- Click `Add` to create the new cantilever, which will appear at the bottom of the cantilever selection list with the name user_cantilever-name. Creating a new cantilever does not effect the selected cantilever. Close the dialog box and select the new cantilever.

The `Edit` tab allows you to change calibration parameters of the selected cantilever.

The `Remove` tab allows you to remove a user-added cantilever from the selection list.

2.2.2.7 Calibrating torsional eigenmodes

Torsional calibration uses the same principals as flexural calibration, but the methods are not as well tested and established as flexural callibration. We provide two methods for calibrating the lowest torsional eigenmode:

- `Torsional: Hydrodynamic Function` uses the analytical expression for the hydrodynamic function of a long rectangular beam, calculated by Green and Sader [Green-2002]. This method calibrates the fundamental eigenmode using the length and width values given in the fields above.
- `Torsional: Flexural + beam theory` requires that you first calibrate the flexural eigenmode. If this is done, the mode stiffness `k_flex` and the resonant frequency `f0_flex` should appear in the fields below. This method uses the ratio of resonant frequencies of the torsional and flexural modes, to calibrate the torsional mode in reference to a calibrated flexural mode. The method is explained in a paper by [Thoren-2017] .

2.2.3 Scanning

Before scanning the host AFM must be configured for ImAFM™ by simply putting it in **contact mode** with the feedback **set-point set to zero volts**. On some AFM's software scripts or special modes are provided to do this automatically. Other AFM's require that you set the break-out box switches in the correct position, so that a 'fake' deflection signal is fed from the MLA™ to the host AFM. The host AFM can then preform scanning feedback on this signal (see *Connection to host AFM*). The feedback parameters, integral and proportional gain, are set in the host AFM software.

The default scanner view has three components: The `Scanner` panel, the `Image Settings` panel and the main view showing the `Amplitude` and `Phase` images. More panels can be added, for example when you want to perform analysis as you scan, making force curves or analyzing transects. These panels will be addressed in the section on *Quantitative Analysis*. Here we describe how to setup and execute the first scan.

2.2.3.1 Setup and Scan

Three controls in the `Scanner` panel specify the parameters for `Setup`. There must be a *Current calibration* in order for the setup to function.

- `Osc. range` is the desired maximum amplitude of oscillation (peak-to-peak) in nanometers of the cantilever when it is free from the surface. When the surface is engaged, the oscillation range will be somewhat reduced, depending on the `Amplitude set point`.
- `Pixel rate (df)` is the spacing Δf between tones in the frequency comb (see *Intermodulation Measurement*). The pixel rate, together with the number of pixels per line determine the `scan rate`.
- `Resolution: x y` sets the number of pixels to acquire in the fast scan direction (x) and the slow scan direction (y). The **x resolution** does not depend on the host AFM setting of pixels per scan line. The x resolution together with the pixel rate determines the `Scan Rate` given by the AFM Suite. The scan rate must be set on the host AFM to the value given by the IMP Suite. The **y resolution** must also be set to the same value on the host AFM, which determines the number of scan lines at the given rate.

`Setup` runs a routine to determine the frequency and amplitude of the two drive tones. You must be well above the surface when you perform setup, several cantilever widths or at least 0.1 mm. If you get an *Out of Range Error* message you may need to adjust the oscillation range or attenuate the input signal (see *Advanced Setup*). This automatic setup is designed for basic ImAFM™ with two drive tones close to resonance. Much more complicated measurements and modulation schemes can be setup with the *Drive Constructor*. When the setup is complete, the `Scan` button will not have gray text, indicating that you are ready to scan.

`Scan` makes the software ready to acquire measurement data. After pressing `Scan`:

- Set the scan rate on the host AFM to the `Scan rate` given in the scanner panel.
- Make sure your host AFM is set to **contact mode** and the set-point is set to **zero volts**.
- Use the host AFM software to engage the sample and start the scan.

When the end-of-line (EOL) triggers are detected, the AFM Suite will begin to collect and display the scan data. After the last scan line when an end-of-frame (EOF) trigger is detected, a scan file is stored (see *Status Banner*). However, the EOL and EOF triggers do not code for the direction of the scan. It may be necessary to sometimes `Flip left-right` or `Flip up-down` to make your image match that displayed on the host AFM.

On some AFM's you can move to the top or bottom of a frame to start a fresh scan without waiting for the current scan to finish. If this action is performed on the host AFM, and if the EOF trigger is sent, the IMP will save a scan file and automatically start collecting a new scan. Some AFM's (notably Asylum) do not send and EOF trigger when you move to the top or bottom of a frame. When the IMP suite is configured for such an AFM, two buttons appear which allow you to `Move to top` or `Move to bottom` to keep the synchronization with the host AFM scan.

During the scan, you can at any time perform the following actions and set the following parameters:

2.2.3.2 Measure just lifted

`Measure just lifted` starts a routine to measure the oscillating cantilever at a point where the tip just stops interacting with the surface. The response at this just-lifted position is needed for *Background force compensation*. Dialog boxes open to explain what the routine is doing. You can choose the `Quick` option to quickly extend the scanner to its full lift position and measure lift, but for accurate measurement you should stop the scan while maintaining engagement with the surface (e.g. temporarily set the scan size to zero) and choose `OK`. The feedback set-point is slowly increased until the total Intermodulation Distortion (IMD) at the surface drops below a target value, determined by the measured IMD at the full lift position (where `Setup` was performed). This full lift IMD is usually just noise as the cantilever response is linear, meaning that there is no IMD far from the surface. When the IMD falls below the target IMD, the response is measured and stored as the lift response. You can `Abort` if something is not working.

If you did not `measure just lifted` during the scan, it is possible to perform background force compensation, if your scan file has a parachuting pixel in the image, as described in *Background force compensation*.

2.2.3.3 Set-point

`Amplitude set-point` is the set-point for the AFM scanning feedback. ImAFM™ feedback is based on the response amplitude at one of the two drive frequencies (Drive 1), and the set-point is given as a percent of the free response amplitude at this drive frequency (see *Setup Feedback*). If you are having trouble engaging the surface, you can lower this set-point. Sometimes background forces cause drop in amplitude by as much as 75% at the surface. This effect can cause a false engage at a position well lifted from the surface. If `Measure just lifted` is performed, amplitude change is calculated as a percent of the amplitude at the just-lifted position. Setting the set-point to more than 100% will cause the probe to lift from the surface – a useful way to lift away from the surface without stopping the scan, if you want to check or adjust something.

2.2.3.4 Scan rate

`Scan rate` is the required rate of scanning, determined by the chosen pixel rate (measurement bandwidth, df) and number of pixels in the scan line (x resolution). You must set the host AFM to scan at the given scan rate, otherwise the images will not be synchronized. Some host AFMs do not allow arbitrary scan rate, but it is not necessary that the scan rates match exactly. A 1% deviation is not noticeable in the images and it is better that the host AFM scan rate be slightly smaller than the ImAFM™ scan rate. You can slow down the scan rate using the same measurement bandwidth and pixel rate, to help track better on a rough surface. `normal` means scanning at the maximum rate and you can choose `half` or `quarter` speed. Making this choice will result in a new calculation of the scan rate which must be set in the host AFM. Slowing down the scan will help to track the surface more closely, making the force measurement more accurate.

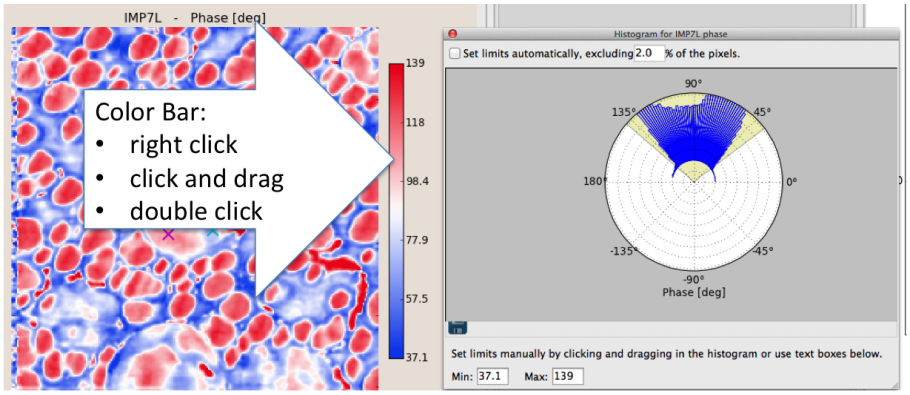
2.2.3.5 Image settings

The panel `Image Settings` contains the controls for displaying which images are plotted

- `IMP Control` lets you select the frequency at which the `Amplitude` and `Phase` images are plotted. The frequencies are ordered from lowest to highest, going from left to right. The text below the slider displays which frequency is being plotted
- `Scan direction` has two buttons which control whether the `Trace` or `Retrace` will be plotted. Data is acquired and stored for both scan directions. Note that both trace and retrace are always stored in every scan. Flipping between trace and retrace can be a good way to see if feedback errors are affecting your image.
- `Swap` will exchange the data stored as trace and retrace. The host AFM trigger signals do not distinguish between different scan directions, and sometimes it is necessary to swap so that trace and retrace are the same as that in the host AFM. Do not worry if you do not get this correct during the scan as it can be easily corrected after the scan session using the *Session Overview*.
- `Flip right-left` and `Flip up-down` do not exchange trace and retrace data. This action can also be done later in the *Session Overview*.
- `Scan size` , when checked, will display axis labels with the image size which must be entered in the `x` and `y` data fields. Units should be given in either nanometers or micrometers, using the characters: nm, um or μm (u will be displayed as μ). Scan size values will be stored when the scan is saved, and these values will be overwritten if and when the scan size is imported together with height data from the host AFM (see *Import host AFM data*).

2.2.3.5.1 Color Bar

The color bars have functionality for adjusting the images:



- **Right click** on the color bar to see a histogram of the plotted values. You can adjust the image contrast by left-click-and-drag on the borders to the shaded region. These borders mark the max. and min. values for the color map used. Data outside the shaded region is forced to either min. or max. If the check box is activated, the software will automatically choose the max. and min. excluding the given percent of outlying values. You can also change the color map in the histogram window.
- **Click-and-drag** upward or downward on the color bar, to adjust the minimum and maximum values respectively. The color value of the center will remain constant. You can do this action with the histogram open.
- **Double click** on the color bar to return to the automatic setting settings.

2.2.3.5.2 Image Toolbar

All plots and images have a toolbar with the following functions:

Home returns to the initial plot or the full image.

Pan-Zoom when selected, left-click-and-drag will cause the plot to zoom, starting from the point of click. Dragging horizontally will zoom only the x-axis, dragging vertically will zoom only the y-axis. Dragging at an angle controls the relative rate of zoom of each axis. Right-click-and-drag will grab the plot at the point of click and slide it in the direction of the drag. Performing these actions while holding down the **x**, **y** or **ctrl** keys will restrict the pan or zoom to occur only in the x-axis, y-axis, or preserving current aspect ratio, respectively.

Zoom when selected, a right-click-and-drag over the plot will zoom to the selected rectangle upon release.

Save image opens a dialog box for saving the image in several formats (png, eps, pdf and more). Sometimes you would like to change the aspect ratio of the plot, or the relative size of the frame and text in your saved image. Simply rescale the entire suite (click-and-drag on the lower right corner), or re-size a particular frame, before you save. This action will rescale the plot and axes while keeping the text and line size fixed.

Configure subplots opens up a dialog box to adjust the placement of the plot axes within the plot frame.

2.2.3.5.3 Mouse wheel zooming

In addition to the zoom tools in the image toolbar, you can use the mouse wheel to rapidly zoom plots and images:

- Hover over the **x** or **y** axis of a plot, just outside the plot frame, and roll the mouse wheel. Only that axis will zoom about the location of the mouse pointer. Click the mouse wheel to auto-scale the axis.
- Hower over any point in a plot or an image and roll the mouse wheel. The plot or image will zoom about that point with equal zoom in the **x** and **y** directions. Click the mouse wheel to auto-scale the image or plot.

2.2.3.6 Selecting Data in a Scan

Two important tools in the image toolbar are the pixel inspector and line inspector tools. These tools do more than simply controlling the plot. They select data at pixels and analyze the data to generate force and parameter plots.

2.2.3.6.1 Pixel inspector tool

A left-click on the icon will activate this tool. When the tool is activated, a left click on either the amplitude or phase image will select the data at the point-of-click, mark it with an **X**, and open the *Signal Inspector* panel. If you have the *Quantitative Analysis* tools installed in your software, the *Force Inspector* will also open and display a force curve. When the pixel inspector tool is active, left-click on the plot will un-select the **X** nearest to the point-of-click and remove the data from the *Data Tree*. The *Quantitative Analysis* tools allow you to analyze the spectral data in many different ways to reveal tip-surface interaction. The *Data Tree* allows you to compare different pixels form the same scan, or different scans, in the same plot.

2.2.3.6.2 Area inspector tool

A left-click on the icon will activate this tool. When the tool is activated, a left click and drag to form a loop in either the amplitude or phase image will select the data inside the loop, marking it a transparent color. All intermodulation spectral data will be averaged in the enclosed area. if you have the *Quantitative Analysis* tools installed in your software, this averaged data will be analyzed as one effective pixel. Averaging over areas with the same response can give much lower noise, and smoother force curves.

2.2.3.6.3 Line inspector tool

A left-click on the icon will activate the tool. When active, a left-click-and-drag on either the amplitude or phase image will select a transect line upon release and a left click will un-select the line nearest to the point-of-click. The `Line Inspector` panel will open and a plot of the amplitude and phase of the currently viewed image will be shown. All data along this line will be selected and available for analysis. If you have the *Quantitative Analysis* tools installed in your software, the line inspector will allow *Analyzing Linear Transects* and plot parameters of the tip-surface interaction along the selected transect.

2.2.3.6.4 Clear pixels tool

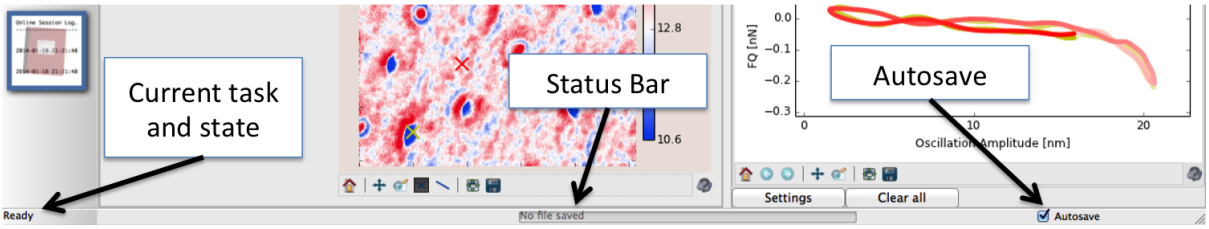
Removes all selected pixels from the image.

2.2.3.6.5 Level image tool

Performs a plane fit to the amplitude and phase image data and subtracts off the plane, thereby flattening the data.

2.2.3.7 Status Banner

The banner at the very bottom of every view has text describing what task the software is currently performing (sweeping, calibrating, scanning, etc.) and a status bar that graphically shows the time required to finish the task. If `Autosave` is checked data is continuously saved to file save when the task is finished. The name of the most recent saved file is also given.



2.2.4 Analyze Scan Data

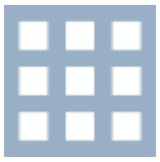
Raw ImAFM™ data, the intermodulation spectrum at each pixel, both trace and retrace, are stored in one compact scan file. The spectrum at each pixel is an highly compressed representation of the actual cantilever motion, from which we reconstruct the tip-surface force. From the raw data, the AFM scientist can go back and make a more careful study of each point on the surface, analyzing it with different models and plotting it in different ways. Intermodulation Products offers a *Quantitative Analysis* package with many methods of analysis, and more are constantly being developed. You can also use the *ScanData* python class to easily extract and analyze the data with your own analysis methods.

In `File` pull-down menu, select `Open (ctrl+O)`. The `Amplitude` and `Phase` images appear in a new tab for each open scan file. You can open multiple tabs and compare data from different scans on the same plot in the analysis view. In the analysis view you will find the *Image settings* panel, the *Image Toolbar* and the *Color Bar*. With the *Pixel inspector tool* and *Line inspector tool*, you can select individual pixels and lines for *Quantitative Analysis*, as previously described.

2.2.4.1 Image Smoothing

The analysis panel is similar to the *Image settings* panel, with an additional `Smooth image` button. This button opens a dialog box where you can apply a Gaussian filter to your scan data. Smoothing will convolve the stored intermodulation data with a Gaussian function of width `Sigma` pixels in the x and y direction. The result is a new image, stored to the given `File name`, where each pixel of the new image is a weighted average of neighboring pixels in the raw scan data file. Smoothing with one pixel results little loss of sharpness in the image and it lowers the noise considerably. The Gaussian has 98% of the weight within 3 Sigma of the center, so smoothing with one pixel essentially averages a block of 9 pixels, giving an improvement in the signal-to-noise ratio by a factor of 3. This is a smart way to improve the signal quality without increasing the measurement time, by exploiting the idea that neighboring pixels are more likely to have the same response. Note however that smoothing does introduce a correlation length to your data, which you can clearly see as a granularity in high-order IMP images, which are often more noisy. Be careful not to interpret these smoothing-induced grains as features in your image.

2.2.5 Session Overview



This view shows all the scans in a session, automatically imports data from the host AFM, sorts out unwanted scans to a sub-folder, adjust images for synchronization problems with the host AFM, and generates parameter maps on multiple scans in a batch processing mode.

2.2.5.1 Session

`Open IMP session` and choose a session folder. An overview is generated showing file info and image thumbnails where each row corresponds to one scan. `Reload` causes the overview to be updated to the current state of the session folder. `View session log` opens a window showing the session log file.

2.2.5.2 Import host AFM data

The AFM height image is a record of the change of the probe height during the x-y scan. The probe height h, is the position of the base or the fixed end, not the position of the tip z at the free end of the flexing cantilever (see drawing in the section *Force Reconstruction Methods*). The probe height is controlled by the scanning feedback and in most modern AFMs the height signal is measured using a linear sensor built in to the scanner. When performing ImAFM™, the height data is recorded by the host AFM. To associate ImAFM™ quantitative analysis with this height data, it is necessary to import the height data in to the IMP Software suite.

`Associate host AFM files`

- `Import height data` : Navigate to the folder on your host AFM computer, network or storage device where the host AFM files are stored. When you OK the selected folder the `Match Images` window opens. Drag the image thumbnails from `Unmatched IMP images` at the bottom, and drop them on the corresponding `AFM images` at the top. When you have matched a few images, click `Match with timestamps` and the computer will attempt to associate all remaining files. If the matching is incorrect, you can drag images away from middle group of `Matched IMP images`. When the matching looks correct, click `OK` and the AFM height data will be appended to the .imp file. (Note: In DI/Veeco Nanoscope software version 5 there is a bug which requires you to simply open and close the file in the DI software before you can import it to the IMP suite.)
- `Hide unchecked` will move all the unchecked scans to a sub-folder of the session folder called 'hidden_scans'. These scans will no longer be displayed in the session overview, but they are not erased. You can always use your computers file browser to move a file from the 'hidden_scans' folder back to the session folder, where it will again be displayed when you click `Reload`. `Unhide all` brings all files from the 'hidden_scans' sub-folder up to the session folder.

2.2.5.3 Flip and swap scans

Because the host AFM does not code the trigger signals for trace and retrace (only for change of fast-scan direction) it may be necessary to left/right flip the scan data or exchange the trace and re-trace data in order to associate the ImAFM™ scan correctly with the host AFM scan. Furthermore, AFM triggers do not code for slow scan direction (only end of frame) so it may be required to exchange up/down flip images. All this is easily done in the Session Overview. To perform these these flip and swap operations you first have to select the scans.

`Checkmarks` control which files will be flipped and swapped: `Check` will select all scans, `Uncheck` deselects all scans, `Toggle` will switch checked to unchecked, and unchecked to checked.

`Apply to checked scans`

- `Flip Left/Right` is performed on both trace and retrace data, but trace and re-trace are not exchanged.
- `Flip Up/Down` is performed on both trace and retrace data, but trace and re-trace are not exchanged.
- `Swap Trace/Retrace` will exchange trace and retrace, with out performing any flip. Note that left-right flip, does not necessarily imply that trace and retrace were stored incorrectly when scanning. Depending on whether you selected the trace or retrace for storing the height data in the host AFM, you may want to exchange trace an retrace to get correct association. ImAFM™ always stores both trace and retrace.

2.2.5.4 Batch process parameter maps and force volume data

`Analyze checked scans` selects which type of *Parameter Maps* will be made on the checked scans in a batch process. As parameter maps are computationally demanding, it is very useful to set up and run batch processes.

- `Parameter Map: Model fit`, when checked, will perform the analysis described in *Model Fit*, generating parameter maps, or color coded images of tip-surface force parameters. The `settings` button opens a panel for choosing the model and fit parameters, as described in *Parameter Maps*.
- `Parameter Map: Polynomial`, when checked, will perform the analysis described in *Fast Polynomial*, based on the polynomial representation of the conservative force-distance curve. The `settings` button opens a dialog with options similar to that for *Parameter Maps*. `Polynomial degree` is described in *Polynomial*.
- `ADFS Force Volume` , when checked, will perform *Amplitude Dependent Force Spectroscopy (ADFS)* on each pixel of an image. The `settings` button opens a dialog with options similar to that for *Parameter Maps*. `Save to HDF5` will create one file containing an ADFS force curve at each pixel, whereas `Save to ASCII` will create a folder with one file for each pixel, where the pixel coordinates are given in the file name.
- `Start` the batch processing for all checked scans. A status bar tells you how far the batch process has progressed. `Abort` will end the process, storing the data analyzed thus far. It is not possible to resume an aborted process.

2.2.5.5 Three dimensional viewer

To generate a three dimensional view you must have imported the height data from your host AFM. **Double click on the height image** to open the `IMP 3D-viewer`. Left click and drag on the image to rotate the viewing angle. Right click and drag to change the zoom.

The `Viewer settings` group allows for change of the 3D image properties, including the `Z scale` which usually defaults to a factor larger than 1, in comparison to the X and Y scales. Such scaling exaggerates the topography.

The `Texture settings` group controls which quantity is represented by the color painted on the topography. Default is the response phase at the first drive frequency. You can choose to plot `Amplitude` or `Phase` of any frequency in the intermodulation spectrum. If the .imp file has an `Add-on image`, for example an ImEFM™ image or a parameter map, these can also be painted on the topography. You can also choose to a `Parameter file` consisting of 2D arrays, the same size as the image, stored as a numpy .npz file. `Limits` opens up a dialog box with a histogram of image pixels, allowing you to set the upper and lower limits of the color scale.

The `Camera position` is shown in the boxes. It is easiest to adjust this by left or right click-and-drag on the image. If the `Animate` box is checked, the image will rotate around the azimuth. You can use screen capture programs to make a movie of this rotation.

The `Plane fit` group has three different methods for flattening the image:

- `None` shows the height data as it is stored in the host AFM height image file.
- `Average` fits a plane over the entire image and offsets all height data so that this plane is flat.
- `Selected points` allows you to flatten on a selected region of the image. With this option selected, hover over the image with the mouse. When you press the 'p' key, the point at the mouse arrow will be selected. After selecting 3 points a plane can be calculated and the image will offset all height data so that this plane is flat. You can continue to select more than 3 points, and the image will be flattened with a best-fit plane to all selected points. The `clear` button removes all selected points. A red frame appears when you select points. To make this frame disappear, hover outside the image and press the 'p' key.

`Save image as...` opens a save dialog so you can save to a desired location, and choose between several different file formats.

`Render movie` will create a movie of a single rotation around the azimuthal angle, as see in the viewer when the `Animate` box is checked. The movie uses 3D ray tracing so it takes a rather long time to generate the movie. This feature is experimental and it requires that you install Povray software.

`Render 3D` will use a 3D ray tracing program to create the same view seen in the viewer, with shadows added. It can take several seconds to render the image. The image will be stored in your session folder, in a sub-folder called `Povray`. This feature is experimental and it requires that you install Povray software.

2.2.6 Session Logbook

The Intermodulation AFM Software Suite keeps a session log while you are working. The software automatically logs all important activity in chronological order with a time stamp. For example: making a new calibration, storing an image, starting or stopping the scan, and much more. This log is a text file formatted for easy reading so you can quickly see exactly what was measured and when it was measured. You can also add your own comments to the log file. Add your comment to text field at the bottom of the panel and finish with `Ctrl+Return`. Your comment will be time stamped in inserted in to the log file.

CHAPTER

THREE

QUANTITATIVE ANALYSIS

The quantitative analysis tools allow you to analyze the measured intermodulation spectral data at each pixel in a variety of ways. Each spectrum is a frequency-domain representation of the tip motion at that pixel, and from this motion we are interested in learning about the forces of interaction between the tip and surface. In dynamic AFM there are several forces giving rise to the cantilever motion: The tip-surface force, cantilever bending forces, viscous drag force on the cantilever body, inertial force due to acceleration of the cantilever mass, and the drive force that we apply by shaking of the cantilever base. To properly separate out the tip-surface force from all other contributions we require three things: A good *Calibration* of the cantilever, and a good measurement of the cantilevers free motion when it is far away from the surface, and if you wish to compensate for background forces, a good measurement of the cantilevers just-lifted motion (see *Measure just lifted*).

Calibration is performed at the outset of ImAFM™ and the free motion is measured during Setup, before scanning. It is easy to lift well clear of the surface and re-do either of these at any time during the session. We recommend that you check the calibration occasionally, to see if the results are consistent with previous measurements. After each re-calibration you also need to re-do Setup and make a new measuremnet of the just-lifted response. The most recently measured or *Current calibration* data, free motion spectrum and lift motion spectrum are stored with each scan file. These measurements are an important part of the analysis of the scan data and they can be viewed using the *Data Tree*.

The focuses in this chapter is on how to use the software to perform different methods of analysis. References are given to the scientific literature where the fundamental physics and mathematics of the force reconstruction methods are explained in detail.

3.1 Inspecting single pixels

The intermodulation spectra at individual pixels can be analyzed by selecting the data with the *Pixel inspector tool*, which puts the data in to the *Data Tree*, allowing you to examine it in the following panels:

3.1.1 Signal Inspector

Activate the *Pixel inspector tool* in the *Image Toolbar*, and left-click on a point in either the `Amplitude` or `Phase` image to select a pixel (right-click to de-select). The data from this pixel is moved to the *Data Tree* and the `Signal Inspector` panel opens where you can see the tip motion durring the selected pixel time. You can display the motion in the following ways:

- `Time` signal shows a very dense plot of fast oscillations with a slowly varying amplitude envelope. Zoom is required to see the individual oscillation cycles.
- `Spectrum` shows only the amplitude of the individual intermodulation frequency components, ploted on a log scale. The component which is plotted with a thicker line corresponds to the amplitude and phase image displayed. You can click on different componets in the plot, and the amplitude and the images will change to the amplitude and phase of that frequency component.

3.1.2 Force Inspector

If the *Quantitative Analysis* software is installed, selecting a pixel with the *Pixel inspector tool* will open the `Force Inspector` panel where you see plots of the analyzed spectral data, giving information about the interaction forces at each pixel. Note that the physical units in the plots are given in calibrated nano-Newtons [nN] (or atto Joules [aJ = nN nm]) versus nanometers [nm], as determined in the *Calibration* step.

The Force Inspector panel has three tabs: `FI FQ Work Force` which control the display of the different different plots as described in detail below. You can control the plots and analysis presented in there three different tabs by opening the *Force Inspector settings* panel.

3.1.2.1 Dynamic force quadratures

The `FI FQ` tab brings up a double plot of two integral quantities, plotted versus the **oscillation amplitude** (not cantilever deflection). The plot $F_I(A)$ is a weighted average of the force which is in-phase with the sinusoidal cantilever motion, determined over a single oscillation cycle of the cantilever. Similarly, the quadrature force $F_Q(A)$ is a weighted average of the force 90 degree phase-shifted from the harmonic motion (in phase with the velocity). It is important to note that $F_I(A)$ and $F_Q(A)$ are **not traditional AFM 'force curves'** . They should not be compared to plots of the instantaneous force vs. distance between the tip and surface. Every single point on the $F_I(A)$ and $F_Q(A)$ plots, represents an integral of the tip-sruface interaction force over one single oscillation cycle (see [Platz-2012b]).

The force quadratures are special in that they are a **direct transformations of the intermodulation spectral data** They are simply another way of looking at the spectral data, without making any assumptions as to the nature of the tip-surface force. They tell us something fundamental about the tip-surface force as it is experienced by the cantilever, in the reference frame of the oscillating cantilever. $F_I(A)$ is a conservative force (e.g. due to elastic interaction) and $F_Q(A)$ is a dissipative force (e.g. resulting from the viscous nature of a moving surface). You can read more about force quadratures and their interpretation in [Haviland-2016].

You may notice hysteresis in the $F_I(A)$ and $F_Q(A)$ curves, where the integrated force on the up-beat (increasing amplitude, lighter shading) is different from that on the down-beat (decreasing amplitude, darker shading). This hysteresis can be related to the finite relaxation time of a visco-elastic surface, or plastic deformation of the surface. However, some measurement artifacts can also cause hysteresis, such background forces that are not properly compensated for (see *Background force compensation*) , over-active feedback (large error signal) or excessive noise in the measured intermodulation spectrum. If the hysteresis is small you can treat it as a weak effect in relation to the overall trend, and in this case you can select `mean curve` when reconstructing the force vs. deflection curve using *Amplitude Dependent Force Spectroscopy (ADFS)*.

3.1.2.2 Dissipated work and average potential

The `Work` tab brings up a double plot very similar to `FI FQ`. We plot the product $2\pi A \times F_Q(A)$ vs. the amplitude A. We can regard this product as a work integral, giving the work done by the tip-surface force during the single oscillation cycle of amplitude A (see [Platz-2012b]). For a purely conservative interaction this work is zero at any amplitude. Negative F_Q means that energy in the cantilever was lost via the tip-sample interaction, for example via motion of a viscoelastic surface, or irreversible deformation of the tip or the sample. In contrast, the quantity $2\pi A \times F_I(A)$ is not a work integral. If the interaction were described by a conservative force, this quantity is related to the average potential energy change in the oscillation cycle with the amplitude A.

3.1.2.3 Reconstructed force

Sometimes refer to data analysis as **inversion** , or **reconstruction** because it involves solving an inverse problem, or reconstructing the force which produced the measured motion, as opposed to forward problem of finding the motion resulting from a given force. As is often the case, the inverse problem is 'ill-posed'. The limited sensitivity of our measurement gives inadequate information for finding a unique solution to the inverse problem. The very small force that we are trying to determine only gives measurable motion at frequencies in a narrow band close to the high-Q cantilever resonance. Our challenge is therefore to reconstruct the force from this partial spectrum, or narrow band response. Fortunately, with enough intermodulation spectral data, and with a few assumptions that physically well motivated, we can find solutions.

The `Force` tab brings up a single plot of the effective conservative force vs. **cantilever deflection**. Zero deflection in this plot corresponds to the equilibrium position, and negative deflection to the cantilever bending toward the surface. In order to reconstruct the force some assumptions about the tip-surface interaction must be made. These assumptions and the different reconstruction methods are described on a separate page:

3.1.2.3.1 Force Reconstruction Methods

Force reconstruction can be performed on individual pixels via the *Force Inspector*, on selected lines in an image by *Analyzing Linear Transects*, or on entire images via *Batch process parameter maps and force volume data*.

ImAFM™ reconstructs tip-surface force $F(d)$ as a function of cantilever deflection $d = z - h$, where the probe height h and the tip position z are measured from a fixed position in the inertial reference frame where the sample is at rest (the lab frame). ImAFM™ force measurement is fast enough such that one can assume constant probe height h at each pixel. This assumption will however break down when the feedback error signal is large and the base is moving rapidly to compensate for a rapid change in surface topography. ImAFM™ makes no assumption about where the surface actually is in relation to the probe height. This is in stark contrast to quasi static force measurement methods, where one must assume a rigid tip and surface when calibrating the measurement of d by moving h. In fact, ImAFM™ allows you to unambiguously *define* the location of the surface relative to h, by associating it with a distinct feature on the $F(d)$ curve (see [Forchheimer-2012]).

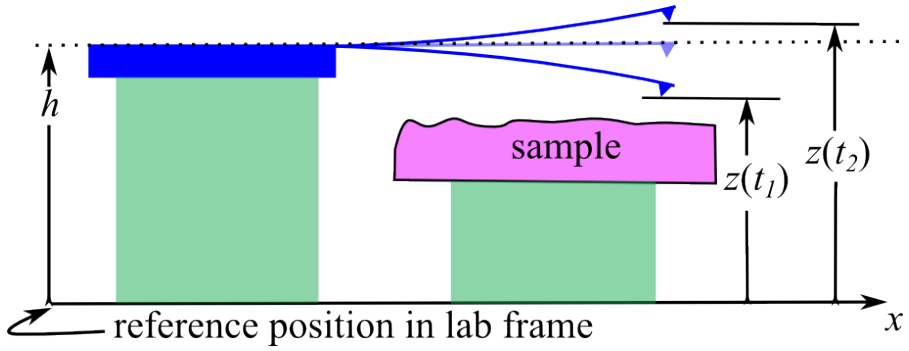


Fig. 1: The AFM detector measures cantilever deflection d. The probe height h is adjusted while scanning. The tip position in the lab frame is $z = h + d$.

Three basic methods are available for reconstructing $F(d)$.

In the `Force` tab of the `Force inspector settings` panel, check the box to activate the desired `Reconstruction Method`. You can use any combination of methods, and plot all results on the same plot. Each method makes different assumptions described below. Click the method name to highlight it and view the options for that method.

Polynomial

The `Polynomial` method approximates the conservative tip-surface force (a function of tip-surface separation only) as a polynomial of finite degree N in the cantilever deflection, d.

$$F(d) = \sum_{j=1}^{N} g_j d^j$$

A further assumption is that the force is zero when the tip is sufficiently far from the surface. Under these assumptions we can find the N polynomial coefficients g_j, both odd and even coefficients, from the measured spectrum of odd-order intermodulation products. The method is described in detail in [Platz-2012a], [Platz-2013b].

`Polynomial` reconstruction has the following options:

- `line style` gives some plotting options for line type and thickness.
- `Maximum IMP order` is largest order of intermodulation product used in the determination of the polynomial coefficients. High order intermodulation products have lower signal level and depending on your scanning conditions, they will disappear into the noise at some maximum order. The maximum order should

be set to the largest order IMP, which has a reasonable signal-to-noise ratio (SNR). You can judge the signal the SNR by simply looking for contrast in the amplitude and phase images of any order IMP.

- `Polynomial degree` is the number of coefficients N in the polynomial approximation of the force. This number can not be greater than the maximum order intermodulation product in your spectrum.
- `Assume localized force` check-box activates the routine to determine the even polynomial coefficients. `Zero force region` sets the crossover deflection d_0, above which the tip-surface force is assumed to be zero. The range is given in normalized coordinates, where -1.0 means closest to the surface, and +1.0 means furthest from the surface. If you uncheck this fitting option you will reconstruct a force curve which is an odd function of deflection, $F(d < 0) = -F(d > 0)$. The odd nature of the curve is due to the fact that the intermodulation spectrum of odd-order products, is only able to reconstruct the polynomial coefficients, g_j with odd j (see [Platz-2013b]). If this odd $F(d)$ curve shows a well developed zero force region for $d > d_0$, then we can safely neglect long-range forces and the assumption that $F(d) = 0$ for $d > d_0$ is well motivated. Move the slider to change d_0.

Model Fit

The `Model Fit` method assumes that the tip surface force is described by a particular interaction model, which has some number of parameters. The method performs a numerical optimization, adjusting the parameters of the model, so as to best reproduce the measured frequency components of the tip-surface force. The method is described in detail in [Forchheimer-2012], [Forchheimer-2013], [Platz-2012a]. Nearly any model can be fit to the experimental data, so long as it can be coded on a computer. However, the solver will spit out a solution and the software will give you a plot which looks like your force model - even if it is a lousy fit to your data. Model Fit should always be used together with Polynomial or ADFS, in order to independently check if the model makes sense. Model Fit also requires reasonable initial conditions for the solver to converge to a good solution.

`Model Fit` reconstruction has the following options:

- `line style` gives some plotting options for line type and thickness.
- `Choose model:` selects the different force models that are programed in the software. You can also add your own models to the software, as described in the advanced section on *Programming your own Force Models*. The help icon opens a window showing the functional form, a list of parameters, and plot of $F(d)$ for each model.
- The table gives a listing of the parameters for each model, and the initial value for the numerical solver. If the check-box is unchecked, the solver will not adjust that parameter and it will be fixed at it's initial value.

`Model Fit` can be applied to every pixel in a scan to generate a parameter map, as described in *Batch process parameter maps and force volume data*.

Amplitude Dependent Force Spectroscopy (ADFS)

`ADFS` works from the assumption that it the the force can be written function of the tip position, i.e. there exists a force curve $F(d)$. It further assumes that the force is zero for positive deflection, i.e. $F(d) = 0$ for $d > 0$. If these assumptions are valid, the $F(d)$ curve is extracted by performing an inverse Able Transform on the $F_I(A)$ curve. The method is described in detail in [Platz-2012b] , [Platz-2013a]. This method can be applied to create a force volume data set from a scan data file, as described in *Batch process parameter maps and force volume data*.

Three features control ADFS:

- `Reconstruct from FI(A) for:` is a pull-down menu selector that allows you to select different regions of the $F_I(A)$ curve that will be used for ADFS. Usually best results are found for `longest monotonous amplitude increase` which is the default selection. However, you can reconstruct for decreasing amplitude, or for a mean curve, which averages the increasing and decreasing amplitude. Note that often with soft materials, the $F_I(A)$ and $F_Q(A)$ show hysteresis, meaning that they are different for increasing or decreasing amplitude. This can be a sign that the surface is not relaxing to its equilibrium position between successive taps of the tip. In this case a force curve $F(d)$ can not describe the interaction, and ADFS gives only an approximate view of the force.

- `Stavitzky - Golay smoothing` is a routine to smooth the ADFS force curves. Smoothing supresses wiggles in the force curves, caused by noise and limited spectral range of the cantilever. Two numbers control the smoothing: The `smoothing window length [%]` is the precent of the total number of points in the full force curve, over which the smoothing is made. You can also choose the `Smoothing polynomial degree`. The smoothing fits a polynomial over a sliding window, to generate a smoothed point.

Fast Polynomial

A computationally efficient method to determine the polynomial coefficients of the conservative tip-surface force is described in [Platz-2013b]. The method is very fast compared to standard *Polynomial* and *Model Fit*. The faster computation enables parameter mapping in real time, as scan data is acquired.

Live parameter mapping available as a script the pull-down menu: Advanced -> Scripts -> `LiveParameterMap`.

- `Start` will activate this feature and you can see the selected `Parameter` as a color map, painted during the scan.
- `Show force curve` when activated will plot the polynomial force curve at the center pixel of every scan line. The force plot displays the selected parameter with a line or point on the force curve.

This type of parameter map can also be applied to scan data file, as described in *Batch process parameter maps and force volume data*.

3.1.2.4 Force Inspector settings

The `Settings` button in the Force Inspector panel opens a separate panel where you find corresponding tabs to control the different plots and the compensation for background forces. You can also open this panel from the Advanced pull-down menue.

- `FI FQ` and `Work` tabs have options to `Use shading` which controls the plotting of the $F_I(A)$ and $F_Q(A)$ curves. When checked (default case) the lightest shade corresponds to the start of the pixel, and the darker shade the end. Use the *Signal Inspector* and select the `Time` tab to see the beat-cycle for the selected pixel. With the standard setup, each pixel starts at close to zero amplitude, with increasing amplitude followed by decreasing amplitude. Thus the shading allows you to see how $F_I(A)$ and $F_Q(A)$ differ from increasing to decreasing amplitude.
- `Force` tab contains several options, explained in detail in *Force Reconstruction Methods*.

3.1.2.5 Background force compensation

The `BG Comp` tab in the Force Inspector Settings controls background force compensation. The compensation requires measurement of the just-lifted response (see *Measure just lifted*). If there is not a just-lifted pixel measured, you will get the same result as if None were selected. The theory and application of background force compensation is described in [Borgani-2017].

- `None` turns off the compensation, in which case all force curves will include background forces.
- `Polynomial` fits the linear response function of the background forces to a polynomial in frequency, of the `Polynomial degree` given. For standard two-drive-frequency ImAFM™, you want to choose the default value polynomial degree = 1 (two points can only determine a straight line). The polynomial of higher degree is only interesting if you use the *Drive Constructor* to create drive schemes with several drive tones.
- `Harm. Osc.` describes the effect of the background forces as a renormalization of the cantilever parameters f0 and Q.

Note: If you did not make a measurement of the just-lifted response during the scan, it is still possible to perform background compensation. If the scan file has a parachuting pixel somewhere in the image you can use the

response at this pixel as the just-lifted response. Select this pixel with the *Pixel inspector tool*. Open the *Data Tree* and find the selected pixel in the tree. Right click on that pixel in the tree and choose `Redefine as lift oscillation`. This redefinition is only temporary. To make it permanent, you must save the .imp file (right click on the file in the tree) possibly with a new file name.

3.2 Analyzing lines and surfaces

At one pixel we can analyze data and make a curve of force or interaction energies as a function of cantilever deflection or oscillation amplitude. These curves can be parametrized, for example by fitting the data to a particular model of the interaction force. It is often very interesting to plot how the parameters change along a linear transect of the image. This type of analysis can be made in real time with the *Line inspector tool*. You can also run analysis of all pixels in a scan file to produce a color map image of each parameter. This later analysis is time consuming and it can be run in a batch processing mode. The creation of these different types of parameter plots and maps are described in the following sections:

3.2.1 Analyzing Linear Transects

If the *Quantitative Analysis* software is installed, selecting a line with the *Line inspector tool* activated will open the `Line Inspector` panel. You will see a plot the amplitude and phase the intermodulation product corresponding to the image showing when the line was selected. All data in each pixel along the transect line is selected for analysis with the *Model Fit* method of force reconstruction.

- The `Line Inspector` panel gives a list of all selected lines in the left column. The color of the word `line` in the list corresponds to the color of the linear transect in the `Amplitude` and `Phase` images.
- Right-click on the word `line` and choose `Fit model` from the menue. The currently active model will be fit to the intermodulation spectral data along the transect line, and a plot will be generated showing how each parameter of the model changes along the transect. Depending on the length of the line and the speed of your computer, there may be some delay while the fitting is being performed at each pixel.
- You can automatically perform the fitting directly after the line is selected by checking the `Fit model` check box at the bottom of the list of lines.
- `Settings` will open the `Line Inspector Settings` panel where you can activated different force models, specify which parameters to adjust, and set the intial values of all parameters, as described in the *Model Fit* section.
- You can test different models along the same linear transect: Simply change the model in the `Line Inspector Settings` panel, then right-click on the word `line` and choose `Fit Model`. A fresh plot of the parameters of the new force model will be made along this line. A new item is added to the list with each fit that your perform. You can always go back to your previous plots by **double-clicking** on that item in the list.
- Right-click on any an item in the list and chose `Save` to export the results of that fit to a text file. You can also save the plot image by clicking the save icon in the *Image Toolbar*.
- Right-click on any item and select `Remove` to take it away from the list.
- Click on the small triangles to expand and contract the list of fits for each line.
- Right-click on either the `Amplitude` or `Phase` image with the *Line inspector tool* activated to **remove** the line nearest to the right-click.

3.2.2 Parameter Maps

You can fit a force model to every pixel of a scan to create a parameter map, or color-coded image of the parameter values. From the `Advanced` pull-down menu, select `Parameter map` to open the `Parameter map creator`. In this panel you may select the `Model` to fit to the data, and the initial values of all parametres, as

described in the *Model Fit* section. Because the generation of parameter maps using *Model Fit* is computationally intensive, it is often easire to use this interface off-line, so as not to overload the computer while scanning. Parameter maps can be generated while scanning using a method based on *Fast Polynomial*. You may PPLY either method to your scan data files from the *Batch process parameter maps and force volume data*, where you can also display the maps as 3D images projected on to the height data.

Analysis is controlled with the following options:

- `scan direction` allows you to choose either the trace or retrace data for analysis.
- `CPU processes` controls how many cores are used when multi-threading the fitting. Fitting a large number of pixels will load the CPU and it may be too sluggish to use while scanning. Adjust this parameter to find a good balance between time-to-output and response time of your computer.
- `X-Y coarse-graining` controls THE number of fits to be performed. If this factor is set to n , fitting will be performed on blocks of $n \times n$ pixels. If it is set to 1, all pixels will be analyzed. Increasing this factor reduces the number of calculations and thus the time to output by a factor n^2, and it results coarse-grained map.
- At the bottom is a list file names that are open in the *Analyze Scan Data* view. All checked files will be analyzed, with the same model and initial conditions. The bar below each file indicates the status of the analysis.
- click `Start` to begin the analysis and `Abort` to end the analysis. The results of the analysis (up to the time of Abort) will be saved in the same folder as the scan file that is being analyzed.

When the model fit is performed on a the scan file **scan01234.imp**, several new fiels are created with the same filename. The files **scan01234_p_name.png** are image files with color-coded images of the parameter values. Such a file is created for each parameter **p_name** in your model. The file **scan01234.npz** contains the numerical data of the parameter values stored in one numpy array. This file can be loaded in to a Python script with the numpy function numpy.load(file_name). Here is an example Python script for reading, and replotting the parameter values:

```
# load the npz file in to the object pmap
pmap = np.load('scan01234.npz')
# extract and plot data for parampater 3, vmin and vmax are color bar limits
imshow(pmap['data'][3],vmin=0,vmax=5)
# display the colorbar
plt.colorbar()
```

3.3 Special Modes

The intermodulaiton measurement concept can be applied to the many different modes of AFM and several special modes are under development. At present the following modes are available.

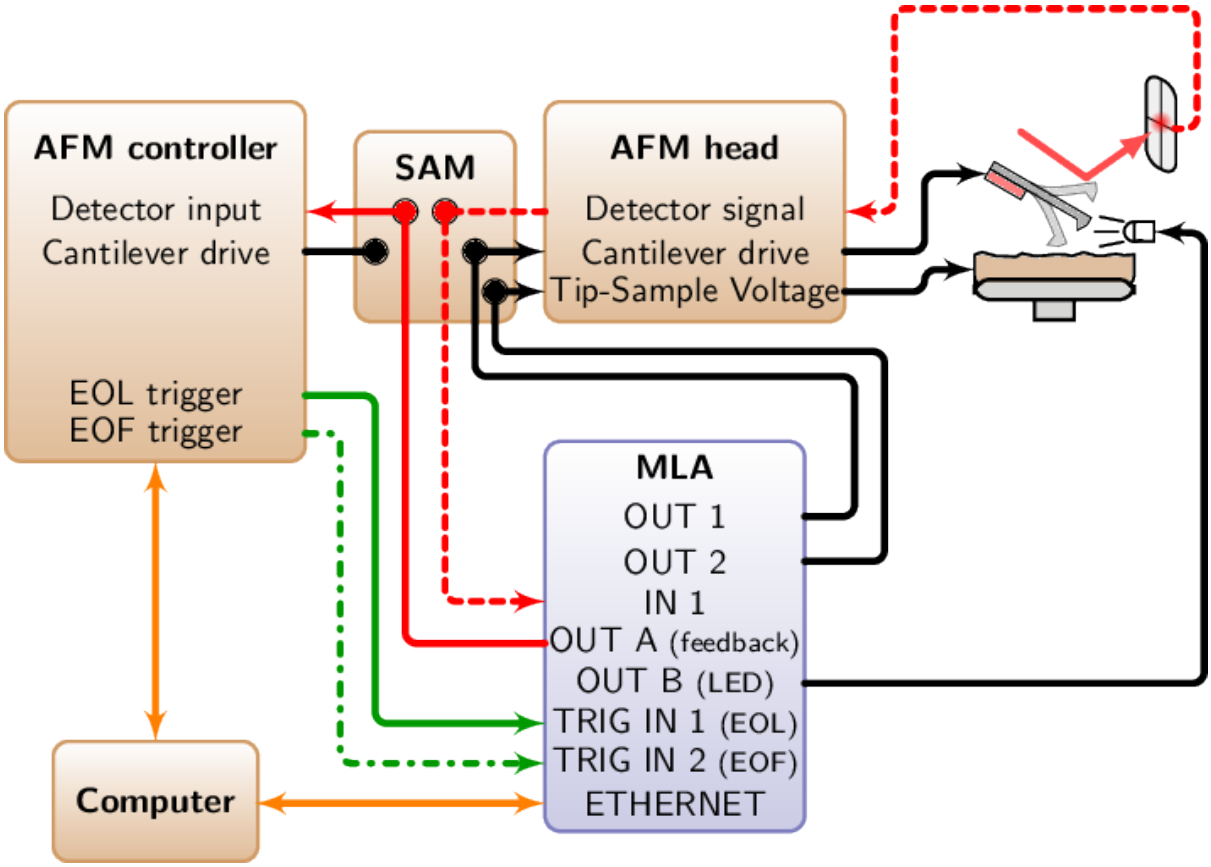
3.3.1 Intermodulation Electrostatic Force Microscopy

The Intermodulation measurement concept can be applied to Electrostatic Force Microscopy (EFM), were a force on the cantilever is induced by a voltage applied between the tip and sample. We call this method ImEFM™, and its implementation requires the synchronous multi-port capability of the *Multifrequency Lockin Amplifier* (MLA™). Two output ports are used with ImEFM™, one port driving the tip-sample voltage at a low frequency and the other port driving the cantilever via the shaker piezo near the cantilever resonance frequency. The tip-surface capacitance gradient is a nonlinear function of the tip-surface separation, and this nonlinearity causes intermodulation or frequency mixing between the electrostatic drive and the mechanical drive. Measuring 4 intermodulation products near resonance and taking ratios of their amplitudes, we extract the contact potential difference between the tip and surface. The theory of the method and its validation was given by Borgani et al. [Borgani-2014]. Here we describe the user interface in the AFM Software Suite.

ImEFM™ requires a conducting cantilever, for example a Pt coating on the front side of the probe. In the description given below, a voltage is applied to the tip, and the sample is grounded. One can equally well apply the

voltage to the sample, and put the tip at at ground. Which method is more convenient to use depends on the host AFM. If you are using a Bruker NSV controller and Icon AFM, the voltage that is applied between the tip and the sample comes from the MLA™ via the composite cable connected to the IMP Signal Access Module (SAM). Make sure that all switches on the SAM are in the IMP position and that the Bruker software is configured to apply voltage to the tip. In the Bruker software, enable advanced parameters (click red puzzle icon) and choose the option **tip bias**, on the setting tip bias. This setting should be effected if the ImAFM workspace is chosen in the Bruker software.

For other host AFMs, you need to work through the signal path for the tip-sample voltage which is applied from BNC port `OUT 2` on the front panel of the MLA™. It can be a good idea to use a BNC Tee and look at this voltage with an oscilloscope in parallel, so that you can actually see the applied signal. A general connection scheme is given below.



3.3.1.1 Setting up the measurement

Before you begin ImEFM™ you must first do the standard calibration procedure in the first two steps of the ImAFM™ *Work Flow*: Finding the cantilever resonance with a *Frequency Sweep* and calibrating the cantilever using non-invasive thermal noise *Calibration*.

Open the ImEFM™ panel by clicking on the ImEFM™ icon in the work flow, or by selecting `ImEFM` from the `Advanced` pull-down menu. The ImEFM™ panel has the following controls:

`Settings` group controls the following:

- `Osc. amplitude` is the desired amplitude of cantilever oscillation. When Setup is executed, the piezo drive voltage will be adjusted to achieve this value. If you are using a standard EFM cantilever, for example

the HQ:NSC15/Pt by micromash (f_0 = 325 kHz, k = 40 N/m , Pt front-side coated) you may want to use about 40-60 nm oscillation amplitude.

- `Tip AC voltage` sets the amplitude of the low-freqeucy AC voltage applied between the tip and sample. This voltage induces a small electrostatic force on the cantilever, which modulates of the cantilever oscillation. The larger this voltage, the larger the electrostatic force. A typical value here is between 1 and 6 Volts. The maximum voltage that the MLA™ can apply is 7 Volts (14 volts by changing an internal jumper).
- `Tip DC offset` sets a DC offset voltage between the tip and sample. Often this can be set to zero, but a small offset is useful when the contact actual potential is close to zero, so that you measure some signal and not only noise. You can also test for a change of potential by changing this DC offset.
- `Set-point` is the fraction of free oscillation amplitude (in %) used as the set-point for the scanning feedback. Setting this this to a smaller value will cause the probe to oscillate closer to the surface. To measure purely electrostatic force the tip should not come too close to the surface, where other surface forces are much stronger than the electrostatic force. A typical value for this parameter is 85% to 95%, however you may want to use a smaller set-point if the free oscillation is measured very far from the surface.
- `Pixel rate` sets the measurement bandwidth, or inverse the measurement time at each image pixel. Making this smaller will improve the signal-to-noise ratio, but it will also slow down the scan rate for a given image resolution (number of pixels).
- `Setup` executes an automatic routine to configure the MLA™ and software for the measurement. After setup is executed, the AC voltage will be applied to the tip.
- `ON/OFF` check box controls the tip-sample voltage. Checked means ON. Uncheck and Check to see if anything changes in your image. If not, you may have a loose or incomplete electrical connection to the tip. If you are using and ImEFM™ with front-panel connections, the tip-sample voltage is applied from BNC port `OUT 2`. You can use a BNC T and oscilloscope in parallel to monitor this voltage.
- `LED toggle` check box controls the LED illumination when measuring surface photo voltage. When unchecked, there will be no illumination on either trace or retrace. When checked the LED will be on during trace, and off during retrace. You can check that this is actually the case by storing the LED voltage as a axillary signal (image) of your host AFM has this capability. The LED voltage is applied from the MLA™ through port `OUT B`.
- `Scheme` selector allows you to choose between:
  - `ImEFM` - the standard mode for measuring contact potential difference. In this mode the measurement bandwidth is equal to the frequency of the applied tip-sample AC voltage.
  - `ImEFM_slow` - scans slower, with better SNR. In this mode the measurement bandwidth is half the frequency of the applied tip-sample AC voltage.

After all settings have been adjusted, press `Setup` to configure measurement, press `Start` , and then use the host AFM software to engage the sample. The system will start to process triggers collect data and build an ImEFM™ image. You must manually set the `Scan Rate` on the host AFM to the value given by the ImEFM™ software, at the bottom of the panel. After you engage and get a stable scan, it is a good idea to press `Measure Free` to make a new measurement of the free oscillation, well above the sample. The electrostatic force is determined from the difference between the engaged oscillation and the free oscillation and accurate reconstruction requires a good measurement of the free oscillation.

The intermodulation spectral data will be processed in real time as you scan so that you can view images of various quantities. All raw intermodulation data and the processed image data will be stored in a the standard file type (e.g. scan01234.imp). You can view previous scans by pressing the `Load .imp` button and navigating to the file.

3.3.1.2 Analyzing and viewing the ImEFM data

`View` group controls how the intermodulaiton spectral data is analyzed and what data is plotted. You can select the following quantities to be shown in the image:

- `Vcpd [V]` - the contact potential difference in Volts.
- `Cz [F/m]` - the tip-surface capacitance gradient $\frac{\partial C}{\partial z}$ in F/m.

- `Czs [F/m2]` - the second derivative of the capacitance with respect to separation $\frac{\partial^2 C}{\partial z^2}$ in F/m^2.
- `Trace` or `Retrace` controls which scan to view. The `Difference` between trace and retrace of `Vcpd [V]` gives the surface photo voltage, when the `LED toggle` is activated (i.e. change in contact potential between light and dark states).
- `Horiz. shift` is the number of pixels to shift to compensate for offset between trace and retrace. This offset arises if the scan overshoot factor or scan speed is not exactly correct.
- `Vert. shift` is the number of pixels to shift to compensate for offset between trace and retrace. This should always be zero, but there maybe something funny with the host AFM.
- `Swap diff.` changes the sign of the difference, i.e from (trace-retrace), to (retace-trace).
- `Cz fit` opens a dialog box `Cz fit settings` that allows you to control the intermodulation frequencies an polynomial order used for calculating the capacitance gradient $\frac{\partial C(z)}{\partial z}$, where a polynomial approximation of $C(z)$ is assumed (see [Borgani-2014]). You can see the result of the calculation at any pixel by clicking the blue pixel inspector icon in the image tool bar. The red pixel inspector icon clears all selected pixels. This feature is in the experimental stage of development.
- `Load .imp` will load a previous scan file with saved ImEFM™ data. If you load a file where ImEFM™ was not performed, you will get a warning, but your EFM images will show something that is not at all related to the electrostatic force.
- `Save .imp` will save the currently analyzed raw data and EFM images. If `Autosave` is checked, each scan will automatically be saved upon completion.

`Debug` group contains some tools for trouble-shooting the measurement, checking to see if all connections are OK, and that you are really measuring an electrostatic force.

- `Set voltage` When checked, the Setup routine will apply the AC voltage between tip and sample. When unchecked, the tip-sample voltage will not be applied and it will be held at ground. In this case, you are simply doing single-freqeuncy dynamic AFM.
- `Use Free` When checked the free oscillation will be subtracted from the engaged oscillation to calculate the electrostatic force. The checked state is the normal and correct mode of operation. When unchecked, the free oscillation is not subtracted. You can un-check and check this box to test if something may be corrupt with your measurement of the free oscillation.
- `Down-sample` will skip the given number of pixels in both X and Y directions, reducing the number of calculations for ImEFM. You may find the need to down-sample if you have a slow computer and you are scanning with high speed and high pixel density. In any case, all raw data will be saved to file and a full resolution image can be calculated offline.

CHAPTER

FOUR

ADVANCED TOPICS

In order to comprehend some of the more advanced features in the AFM Suite, it is necessary to understand in more detail how *Intermodulation Measurement* and *Noise Calibration* work. After this general discussion we continue with sections devoted *Programming your own Force Models*. We explain how to use the the *Data Tree* which organizes the analysis and allows you to view spectral data and extract it to an external text file. A section is included on *File Management*, describing the different file types and where they are stored in the computer. We describe how to move and re-arange the different *Panels and Views* and create your own custom screen layout.

Other advanced sections are given for the AFM scientist who want to experiment with different types of multifrequency measurement and analysis. The AFM Suite has an *Advanced Setup* panel and a *Drive Constructor* to configure the drive tones and measurement frequencies of the Multifrequency lockin Amplifier (MLA) ™. A *Stream Recorder* is also included to record continuous streams of intermodulation data for analysis with your own software. A *Scripting Interface* is also described which allows you to fully customize your multifrequency measurements with scripts that run in the IMP suite. We also document the ScanData python class which organizes the multi-frequncy data. With the functions in ScanData you can write Python scripts that analyze an plot the data stored in the scan files (e.g. scan01234.imp).

4.1 Intermodulation Measurement

Intermodulation AFM (ImAFM™) would not be possible without *Multifrequency Lockin Amplifier* (MLA™) also known as the Intermodulation Lockin Analyzer (ImLA™) [Tholen-2011]. The instrument is specially designed to make a phase-sensitive measurement of many intermodulation products generated by a nonlinear system that is driven by two or more tones.

Intermodulation should not occur when the cantilever is oscillating in air above a surface because the freely oscillating cantilever is a linear system. When the cantilever is oscillating close to a surface and the tip is interacting with the surface the cantilever oscillation becomes nonlinear, causing the generation of intermodulation products of the two drive tones. Measuring both the amplitude and phase of the intermodulation products allows one to reconstruct the nonlinearity, or the tip-surface force. Different force reconstruction methods are described in the chapter on *Quantitative Analysis*.

The intermodulation measurement concept can be extended to multiple drive tones and the MLA™ can be configured to drive with more than two tones. A special *Drive Constructor* interface is provided for setting up the multi-frequency drive and configuring the MLA™ to measure at the desired frequencies. Through a process called tuning, the MLA™ and its accompanying software force all drive frequencies to be integer multiples (or very nearly integer multiples) of one base frequency Δf and this base frequency serves as a reference signal for lockin measurement of the response. If all the drive tones are integer multiples of the base frequency, all of their intermodulation products will also be at integer multiples of the base frequency.

Such multi-frequency signals are called **frequency combs** because their frequency representation, when plotted as a power spectrum, looks like the teeth of a hair comb, with sharp spectral lines equally separated by the base frequency Δf. In the time domain the frequency comb is simply a periodic waveform with period $T = 1/\Delta f$. A general statement of the intermodulation measurement concept is the following: Drive a system with a frequency comb and measure a response comb. The nonlinearity can be reconstructed from the difference between the response and drive comb. The underlying assumption is that the response is periodic on the long time scale T associated with the base tone.

The MLA™ has two different modes:

4.1.1 Time mode

`Time mode` sends all samples to the computer, which can be a huge amount of data if the measurement time is long. Therefore, it is often favorable to down-sample the response, averaging sequential samples to give an effective lower sampling frequency. Down-sampling reduces the noise and it allows for the transfer to the computer of longer, uninterrupted streams (see *Stream Recorder*), but it does reduce the maximum frequency that an be analyzed. A DFT of a time mode data stream will give you the entire intermodulation spectrum, up to the Nyquist frequency, or half the sampling frequency. This spectrum reveals all the mixing in your non-linear system. When finding intermodulation products in ths manner, there is a latency between the end of the measurement time window $T_m = 1/\delta f$ and finish of the calculation of the Fourier coefficients, which is the time require to DFT the window of sampled data. Even when using the Fast Fourier Transform (FFT) algorithm to execute the DFT, this latency is too costly when scanning over a surface. In such situations one uses the MLA™ in frequency mode.

4.1.2 Frequency mode

`Frequency mode` does the Fourier Analysis of many frequencies in parallel inside the MLA™ as the samples are acquired. In frequency mode you get the full bandwidth of the MLA™ up to the Nyquist frequency set by the sampling freqeuncy of the A/D converters in the MLA. The MLA™ performs 2N Fourier sums in real-time at user-specified frequencies, where N is the number of tones in your version or the firmware. These sums give you the Fourier cosine and sine coefficients, or the in-phase and quadrature response. This response can be converted to amplitude and phase with some calculation, but often the analysis of data is more conveniently done on complex numbers.

4.1.3 Feedback

The MLA™ has a feature specifically designed for AFM, where feedback is required to track the surface when scanning. The amplitude of response at one dedicated feedback frequency is calculated inside the MLA™. This amplitude is subtracted from a set-point amplitude, forming an error signal. A voltage proportional to this error signal is sent to one of the output ports. In order to make the feedback more responsive, the MLA™ updates the error signal 250 times in the measurement time window $T_m = 1/\delta f$. The sections on *Setup Feedback* and *Drive Constructor* give more information on how the configure the feedback and set-point.

4.2 Noise Calibration

A quantitative measurement of tip-surface force requires an accurate calibration of the cantilever. Noise calibration uses the enhanced force sensitivity of a high Q resonance to measure the thermal equilibrium fluctuations of cantilever deflection, and thereby extract the calibration constants of both the cantilever and the detector. The method models the cantilever's free dynamics as a simple harmonic oscillator; an accurate model for one cantilever eigenmode, in a frequency band near resonance.

ImAFM™ also takes advantage of this enhanced sensitivity and high accuracy, using only signals near resonance to reconstruct the force between the tip and the surface. Furthermore, ImAFM™ determines tip-surface force as a function of the cantilever deflection at *fixed probe height*. Thus, both axes of the reconstructed force vs. deflection curve (or force-quadrature vs. amplitude curve) rely on one and the same calibration. Quantitative measurement with ImAFM™ is therefore completely independent of the scanner calibration [Platz-2012a] , [Forchheimer-2012]. Noise calibration is performed for each cantilever at the start of the ImAFM™ session and it should be redone if any adjustments are made to the detector. Calibration is easily done at any time during a session so it can be frequently redone to check for consistency. Since the accuracy of quantitative ImAFM™ is traceable to this one measurement, it is important to understand the physical ideas behind noise calibration.

Calibration in the field of AFM is traditionally discussed in terms of finding the spring constant k_s associated with the static (independent of time) deflection of the cantilever. Finding this one constant is sufficient to determine a static force F, which gives rise to the static deflection d,

$$d = \frac{1}{k_s} F$$

ImAFM™ is a dynamic method, where force is determined from cantilever motion $d(t)$. A linear relation between force and motion also exists for dynamic AFM if deflection is small in comparison to the cantilever length. However, due to inertia and damping, the constant of proportionality is frequency dependent. This linear relation between dynamic force and dynamic deflection is most easily expressed in the frequency domain.

$$\hat{d}(\omega) = \frac{1}{k} \hat{G}(\omega) \hat{F}(\omega)$$

The frequency dependence is contained in the dimensionless factor $\hat{G}$, called the transfer gain, and the hat denotes a complex quantity, with the real part being the Fourier cosine component, and the imaginary part the sine component at the angular frequency ω. The complex, frequency-dependent quantity which relates each frequency component of the force to the component of the motion at the same frequency, is called the linear response function, and it is usually denoted $\hat{\chi}(\omega)$. For the simple harmonic motion of each eigenmode of the cantilever,

$$\hat{\chi}(\omega) = \frac{1}{k} \hat{G}(\omega; \omega_0, Q) = \frac{1}{k} [1 + i \frac{\omega}{\omega_0 Q} - \frac{\omega^2}{\omega_0^2}]^{-1}$$

Three parameters are needed to calibrate each eigenmode: The mode stiffness k, effective mass m and viscous damping coefficient η, or alternatively, the stiffness k, resonant frequency $\omega_0 = \sqrt{k/m}$ and the quality factor $Q = \sqrt{mk}/\eta$. A complete calibration requires a fourth constant that converts the detector signal as measured in the digital counting units used by the MLA, or Analog-to-Digital Units (ADU), to the actual cantilever deflection in nanometers. This fourth constant is called the detector responsivity α [ADU/m]. Note that here we could express the responsivity in SI units Volts/m, which would further require a calibration of the MLA against a voltage standard.

All four constants can be determined from one thermal noise measurement as described in [Higgins-2006]. Here we give a somewhat different derivation of the method which is based on the idea that the cantilever is in thermal equilibrium with the damping medium at temperature T, and that the damping can be calculated using hydrodynamic theory. Thermal equilibrium and linear response allow us to use the fluctuation-dissipation theorem to relate the power spectral density of fluctuations of cantilever deflection, to the imaginary part (the dissipative or damping part) of the linear response function.

$$S_{dd}(\omega) = \frac{2k_B T}{\omega} \mathbf{Im}[\hat{\chi}(\omega)] = 2k_B T \eta \frac{1}{k^2} |\hat{G}(\omega)|^2 \text{ [m}^2\text{/Hz]}$$

The simple harmonic motion gives us the functional form of $|\hat{G}(\omega)|^2$. We fit this function to the measured noise peak at resonance, thus determining ω_0 and Q. To get the magnitude of S_{dd} we require knowledge of the damping coefficient η. Hydrodynamic theory is used to calculate the damping due to the surrounding fluid (air, water, etc.) [Sader-1998]. The result can be written as:

$$\eta = L\mu \text{ Re } \Lambda(\text{Re})$$

where L is the length of the beam and $\Lambda(\text{Re})$ is a dimensionless hydrodynamic function of the Reynolds number,

$$\text{Re} = \frac{\rho b^2 \omega_0}{4\mu}$$

The Reynolds number is a dimensionless quantity, which physically corresponds to the ratio of inertial forces to viscous forces in the hydrodynamic flow around the beam. It is formed from the resonant frequency, the fluid density ρ and viscosity μ, and the width of the cantilever b which is the relevant length scale for desribing the flow.

The hydrodynamic function $\Lambda(\text{Re})$ has been calculated for beams of arbitrary cross-section with length much larger than width (either transverse dimension), $L \gg b$ [Sader-1998]. However, a generalization of the theory shows that the basic idea of desribing the damping in terms of a hydrodynamic function of a Reynolds number, applies to an any elastic body of arbitrary shape which is damped by motion in a fliud [Sader-2005]. The beauty of this approach is that once the relevant hydrodynamic function $\Lambda(\text{Re})$ has been determined for a cantilever (or plate) of a particular shape as described by it's plan-view dimensions (effective length and width), we can use this knowledge to calculate the fluctuation force on any cantilever of that shape simply by measuring the resonant frequency in the fluid [Sader-2012].

We measure the total noise which consists of two contributions: the cantilever fluctuations S_{dd}[nm^2/Hz], and the detector noise S_{DD}[ADU^2/Hz]. Since the cantilever noise is statistically independent from the detector noise, we can simply add their noise powers to get the total noise power spectrum at the output of the detector,

$$S_{\text{tot}}(\omega)[\text{ADU}^2/\text{Hz}] = S_{DD} + \alpha^2 S_{dd}(\omega)$$

We fit this expression to the measured data, with α, ω_0, Q and S_{DD} as fitting parameters. Using the value of ω_0 determined from the fit, we can calculate the Reynolds number Re and thereby the hydrodynamic function to determine the damping coefficient η. This coefficient, together with the value of ω_0 and Q determined from the fit, give the mode stiffness k. Thus, thermal equilibrium linear response theory and the hydrodynamic damping theory together give us all quantities needed to determine the linear response function of the cantilever and the detector.

It is important to point out that the hydrodynamic damping function, and thereby the theoretical value of the damping coefficient η, depends on the mode shape and is therefore specific to the particular eigenmode. Calculations show that the mode shape of the fundamental eigenmode of a long rectangular beam is particularly insensitive to the Reynolds number. Higher eigenmode shapes are sensitive to Re [Sader-1998] and the mass of the tip placed at the end of the cantilever. This calibration method is therefore applicable to the fundamental eigenmode and we can expect it to be valid in different fluids, but its applicability to higher eigenmodes is questionable.

We should also stress that reconstructing force from the measured deflection over a wide frequency band, for example from many harmonics of a single drive frequency, requires determination of the detector response function $\hat{\alpha}(\omega)$. An ideal detector should have a flat (frequency-independent) response function up to it's cut-off frequency where it it will start to roll-off, or decrease with frequency to some power. However, real detectors do not always behave in this ideal way, and the detector electronics can introduce frequency-dependent amplitude changes and phase shifts over a wide frequency band. Since ImAFM™ works in a narrow band, we escape this difficulty and we can expect accurate calibration and therefore accurate force reconstruction with the assumption of a frequency-independent detector response function.

4.2.1 Note on sensitivity and accuracy

In the above text we emphasized that **accuracy** is higher when we base force reconstruction on the callibration of high Q resonance. This notion of accuracy is rooted in physical understanding: We are better able to model the force transducer and deflection detector in a narrow frequency band near a single eigenmode resonance, and thus make a good calibration in this narrow band. We also argued that **sensitivity** is higher near resonance. Our notion of sensitivity is independent of that regarding accuracy and it does not originate from theoretical understanding of the physics of cantilever beams or opto-electronic detection. Sensitivity has to do with the signal-to-noise ratio (SNR) of *measurement*.

The field of AFM has unfortunately adopted the term 'inverse optical lever sensitivity' (or invOLS) to describe the constant α^{-1} which converts the measured detector signal (e.g. Volts) to nm of cantilever deflection. We use the term **responsivity** for α because it is actually the magnitude of a linear response function, and it does not tell us anything about the SNR of the measurement. The sensitivity of the measurement is described by two quantities which are displayed in the `Calibration Result` panel.

`Thermal noise force` is the random force due to collisions of the molecules in the damping fluid with the cantilever. This random force has zero average value but the average of the force squared is not zero. The thermal noise force is the square root of the power spectral density of force fluctuations $\sqrt{S_{FF}(\omega)}[\mathrm{fN}/\sqrt{\mathrm{Hz}}]$. The force noise power spectrum is independent of frequency, $S_{FF} = 2k_BT\eta$ and this type of 'white noise' arises whenever the damping force is simply proportional to velocity. With this thermal noise force we can calculate a **minimum detectable force** F_{min}, or the force measured with $SNR = 1$ in a given measurement time T (measurement bandwidth $B = 1/T$).

For example: The scan speed and image resolution dictate a pixel time of $T = 1$ ms, and the thermal noise force is $\sqrt{S_{FF}} = 22$ fN$/\sqrt{\mathrm{Hz}}$. The minimum detectable force at each pixel is therefore $F_{\mathrm{min}} = \sqrt{S_{FF}B} = 696$ fN. This minimum force is the 'signal' which just equals the noise. A good measurement is typically characterized by a signal which is many times the noise level, or $SNR \gg 1$. Note that you must scan 4 times more slowly to decrease F_{min} by a factor of 2.

The detector `Noise floor` is the second quantitative measure of sensitivity. If you measure at frequencies away from the high-Q resonance, SNR is typically not determined by the actual force noise of the cantilever in its damping medium, but rather by the electronic noise of the detector, S_{DD}. We express this detector noise as an *equivalent* deflection noise by dividing it with the detector responsivity, $\sqrt{S_{dd}^{\mathrm{equiv}}} = \alpha^{-1}\sqrt{S_{DD}}$. Here again we assume that the detector noise is independent of frequency in the band of interest. If the pixel time is $T = 1$ ms and $\sqrt{S_{dd}^{\mathrm{equiv}}} = 150$ fm$/\sqrt{\mathrm{Hz}}$, the minimum detectable cantilever deflection at each pixel would be,

$d_{\min} = \sqrt{S_{dd}^{\text{equiv}} B} = 4.7$ pm.

It is also interesting to express the detector noise as an `Equivalent force` noise $\sqrt{S_{FF}^{\text{equiv}}} = k\sqrt{S_{dd}^{\text{equiv}}}$. For the example given above and $k = 40$ N/m, we find $\sqrt{S_{FF}^{\text{equiv}}} = 6$ pN/$\sqrt{\text{Hz}}$, which, in a measurement time $T = 1$ ms gives an equivalent minimum detectable force of $F_{\min}^{\text{equiv}} = kd_{\min} = 0.19$ nN. This would be the minimum detectable force with the cantilever if the measurement were dominated by detector noise, as is often the case for quasi-static force measurements.

Thus, sensitivity of the measurement is characterized by two quantities: The noise force of the cantilever, which represents a fundamental limit, and the noise of the detector. One would like the detector noise to be as low as possible and a very useful way to compare different detectors is to use the ImAFM™ system to perform thermal noise calibration on the same cantilever mounted in two different host AFMs. Since the cantilever force noise is identical in each case, we can compare the ratio of the thermal noise peak, to the detector background noise, or the `Peak to flat ratio`. The larger this ratio, the better your detection. It is also interesting to track this number as you measure. If this ratio becomes smaller, it may mean that the detector is poorly adjusted, or there may be some additional source of detector noise.

4.3 Programming your own Force Models

One powerful feature of ImAFM™ is the ability to test different tip-surface interaction models on the same data set. Data is stored as the raw intermodulation spectrum at each pixel, which is a frequency-domain representation of the tip motion. The *Model Fit* section of *Force Reconstruction Methods* describes how to use the AFM Suite to fit this data to an a force model. This fit can be done at an individual pixel (see *Force Inspector*) , along a liner transect (see *Analyzing Linear Transects*), or on an entire image (see *Parameter Maps* and *Three dimensional viewer*). Several common force models are programed in the IMP Suite, and it is very easy to program your own force model and have it appear in the IMP Suite.

Constructing a good and physically well motivated force model can be difficult. However, once you have a good force model, programing it in the AFM Suite is simple. Some knowledge of the Python language is necessary (see Python). Simply copy the **examples.py** file and give it a new name, e.g. **my_force_model.py** , keeping it in the **force_models** folder. Edit the copy to create your own force model. The **example.py** file is well documented with comment lines to help you understand the idea behind the Python code.

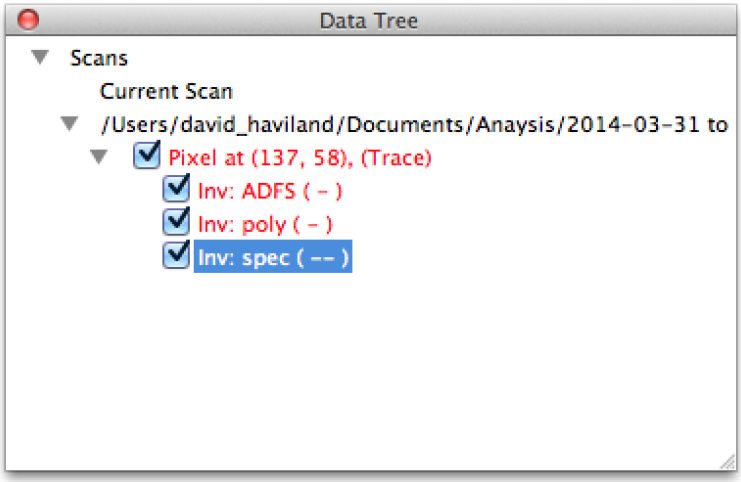
After you have finished with **my_force_model.py**, restart the AFM Suite. Your force model will be automatically loaded and it will appear as an option in the `Model` pull-down menu. Note that any changes you may have made to the **examples.py** file will not be saved, and upon restart, it will be returned to its original, default state.

4.4 Data Tree

The `Data Tree` panel is available from the `Advanced` pull-down menu. The panel can be docked in either the scanning view or the analysis view, or both. This panel lists scans which are open in the analysis panel as well as the current scan. Under each scan there is a list of all the pixels which have been selected with the *Pixel inspector tool*. The color of the text for each pixel corresponds to the color of the cross marking the pixel in the image. Under each pixel there is a list of the different force reconstructions done on that pixel.

Right click on a scan file name and select:

- `Open shell here...` to open a Python shell with an object called `image` containing all the data stored in this scan file. The data id of the scan used by the data tree is given, along with a couple of useful commands. The shell is for advanced users who will need the developer documentation to use this feature effectively.
- `Show meta data` to open a box to view the calibration data stored with this scan file (i.e. the *Current calibration* at the time that the data file was closed) as well as a time and frequency domain plot of the `Free` and `Lfited` motion.



- `Save as ascii...` to open a dialog box where you can save all amplitude and phase data in ASCII format. The save will create individual files in a new, empty folder. There will be one file for both amplitude and phase at each of the measured frequencies, both trace and retrace. For example, when 32 frequencies are measured, 128 data files will be created. If the image is 256 x 256 pixels, this corresponds to about 140Mbytes of data in tab-delimited text, with a carriage return at the end of each scan line.

  The name of each file begins with a number which codes for the frequency. The number corresponds to the index of the 'k-array' which stores the integers multiples of Δf which give the frequency values where amplitude and phase is measured (see *Intermodulation Measurement*). In the folder there is also a file called *metadata.txt* which contains a list of integers in the array *LO_IMP_KARRAY*. The number at the beginning of each file corresponds to the index of this array (zero being the first element). The integer values stored in this array, times *LO_DELTA_F*, gives the frequency at which the amplitude and phase is measured.

- `Save as imp...` to open a dialog box where you can resave the entire scan file with all meta data to a new .imp file. This is usefull when you make changes to the scan data, for example if you redefine the lift oscillation (see below).

Right click on a pixel or patch and select:

- `Open shell here...` to open a Python shell with an object called `pixel` containing the data relevant to this pixel. The data id used by the data tree is given, along with a couple of useful commands. The shell is for advanced users who will need the developer documentation to use this feature effectively.
- `Delete` removes the pixel from the data tree. The cross will also be removed from the image, and the plots from the Fore Inspector panel. You can also remove any pixel by right-clicking close to that pixel in the image when the *Pixel inspector tool* is selected.
- `Save time data to text file` to export the cantilever deflection vs. time during the time window for this pixel. This data is is actually the inverse FFT of the spectral data taken by the MLA™, with some zero padding to give appropriate sampling in the time domain.
- `Save spectrum data to text file` to export create the raw spectral data measured by the MLA™ at this pixel.
- `Save FI FQ data to text file` to export the FI(A) and FQ(A) data for this pixel (see *Dynamic force quadratures*).
- `Redefine as lift oscillation` is very usefull if you have a file with a pixel, or group of pixels (patch) selected with the *Area inspector tool*, where the cantelever has lost contact with the surface. This 'parachuting' pixel or patch can be used to redefine the just-lift oscillation in the .imp file (see *Measure just lifted*). Note that the redefinition is only valid in the loaded scan data. It is not saved to disk unless you actively save a new version of the .imp file.

The check-boxes at each pixel and each force reconstruction controls plotting. Un-check to remove the cross from the image or plot from the Force Inspector panel, without actually removing the data from the data tree. Checking the box will show the plots again.

Pixels selected in the current scan will not be removed when a new scan is generated. When the scan moves passed a selected pixel, the image no longer corresponds to the selected pixel. Selected pixels must be removed manually in the current scan.

`Clear All` in the `Force Inspector` panel removes all selected pixels in all scans.

Right click on a force reconstruction and select:

- `Open shell here...` to open a python shell with an object called `force` containing the result of the force reconstruction at this pixel. The data id used by the data tree is given, along with a couple of useful commands. The shell is for advanced users who will need the developer documentation to use this feature effectively.
- `Get fitted parameters` to open a dialog box showing the fitted parameters when the *Model Fit* method is used.
- `Save Data to text file` to export the force curve found using that reconstruction method.

4.5 Advanced Setup

The `Advanced Setup` view is initially selected in the `Advanced` pull-down menu. Here you can control the drive for standard Intermodulation AFM using two drive tones, set the feedback, and measure the response in real time.

`Requested` fields are the desired, or target values of:

- `f1 [kHz]` the frequency of drive 1
- `f2 [kHz]` the frequency of drive 2
- `df` the measurement bandwidth, df = abs(f2-f1) / n, where n = 1,2,3...
- `A1 [%]` the amplitude of drive 1, in percent of full output
- `A2 [%]` the amplitude of drive 2, in percent of full output

`Actual` show the values that you get, which depends on which algorithm you choose for frequency tuning.

4.5.1 Frequency tuning

Frequency tuning is key to performing measurements with the **Multifrequency Lockin Amplifier (MLA)**™ (MLA™). Tuning is the process of selecting the base tone **df** and the actual drive tones, so that they meet the criteria for ImAFM™; that all frequencies are integer multiples **df** (see *Intermodulation Measurement*). This interface offers three different tuning schemes:

- `None` takes the requested frequencies and does not tune them at all. Without tuning, multi-frequency linear response will be corrupted by Fourier leakage between drive tones and it is not possible to get an accurate measurement of intermodulation.
- `Rounding` selects the closest frequencies to the requested values, such that all frequencies will be near-integer multiples of one base frequency. Rounding gives a very small amount of Fourier leakage between tones, but in almost all cases this leakage is well below the dynamic range of the measurement, and therefore considered negligible. The phase error is corrected at the end of each measurement time (T=1/df), so there should be no noticeable phase drift. With this algorithm there is considerable freedom in the choice of frequencies.

- `Integer` is a more strict algorithm, where the condition of integer multiples is met exactly. This algorithm gives no Fourier leakage between tones and no phase error, but it is more restrictive in the choice of the possible values of df. You should use this algorithm when the MLA™ is synchronized to external frequency generators. To avoid phase drift, you should make sure that the frequency of the external generator is an exact integer multiple of **df** given in the AFM Suite, to as many significant figures as possible with your generator.

The exact properties of these different tuning options have to do with the way in which frequencies are synthesized and analyzed in the MLA™ which we do not discuss in detail here. It suffices to say that the small errors induced in the **Rounding** algorithm are negligible for ImAFM™, and we recommend that you use this tuning. The automatic setup uses this algorithm.

4.5.2 Setup Feedback

Before you engage a sample, and after you have made any changes to the drive configuration, you must setup the feedback. Feedback for ImAFM™ is controlled by the amplitude of response at one of the frequencies measured by the MLA™ (see *Feedback* section). For the basic two-frequency drive, feedback is run at frequency **f1**. With the *Drive Constructor* you have more control over which frequency to set as the feedback frequency.

- `Setup` will adjust the feedback so that zero volts at the feedback output port corresponds to the given set-point.
- `Set-point` is the desired response amplitude, given as a percentage of the free response amplitude. Adjusting the set-point to more than 100% will cause the probe to lift away from the surface. The new set-point values are active as soon as they are entered in the field, and you hit return (or tab).

The feedback port sends out an error signal for PID control of the Z-scanner. This feedback error signal is put back in to the host AFM controller, in the place of the cantilever deflection signal. If the host AFM is set to contact mode with the feedback set-point in the AFM software set at zero volts, the AFM's PID control will adjust the Z piezo to keep the error signal at zero volts, or the value of the set-amplitude. The feedback gains are adjusted in the host AFM which is actually performing the feedback, using the error signal is calculated by the MLA™.

For example: If the set-point is set to 85%, the system will engage the sample until the response amplitude at the feedback frequency drops to 85% of its free value. When this 85% condition is met, the MLA™ will have zero volts at the feedback output. In our experience this amplitude feedback scheme is sufficient for tracking most surfaces. When scanning large areas on rough surfaces, the feedback has to respond rapidly in order to track the surface, and rapid adjustments of the cantilever base height are necessary. Force measurement is very difficult when the cantilever base is rapidly accelerated by an over-active feedback. This acceleration gives rise to additional forces on the tip which are not taken in to account in the reconstruction algorithms. Hence overactive feedback means inaccurate force measurement. Improving feedback and actually incorporating the feedback signal in to the force reconstruction is a subject for future research and development.

4.5.3 Measure

You can measure and plot the response in real time using two different modes:

- `Run time mode` starts the measurement of the response in the *Time mode*, and plots the sampled stream of data in the time domain and its discrete Fourier transform (DFT) in the frequency domain. The discrete data is plotted with a connected blue line. You will notice a jumping of the phase of the oscillation envelope in the time-domain signal. This is normal as the data are analyzed and plotted in asynchronous chunks, and not as one continuous stream. The DFT of the time-sampled data is however stable as each chunk as the same sampling frequency and same number of samples.
- `Run freq. mode` starts the measurement of the response in the *Frequency mode* and plots the amplitude at the measured frequencies (vertical red bars) in the frequency domain, and the oscillation envelope in the time domain (connected red lines)
- `Stop` will terminate the measurement. You must stop the measurement before you start other measurements, for example scanning a sample or taking stream data.

4.5.4 Units

You can select between displaying the cantilever response signal in Analog-to-Digital Units `ADU` , or in nanometers `nm`. You must have a *Current calibration* in order to display in nanometers.

4.5.5 Manual Setup

The MLA™ allows for driving with more complex drive waveforms involving more than two tones, and measuring with lockin schemes that do not simply measure closely spaced tones around one resonance. The AFM Suite offers a few tools to aid in configuring the MLA™ for such measurements. Select `Manual setup` from the Advanced pull-down menu. You can dock this panel, for example in the Advanced Setup frame.

- Select a `Measurement time window` or `Measurement bandwidth`. These quantities are the inverse of one another, and when you change only one of these, the system will calculate the other to match your change.
- Input the frequencies of all tones where you either want to drive, or listen for response. These frequencies can be given as `Freq. kHz`, or as `n` , the later being integer multiples of the measurement bandwidth (see *Intermodulation Measurement*).
- Set the `Amplitude` and `Phase` of the tones only where you wish to drive. If the amplitude is set to zero, there is obviously no drive at that frequency. Setting Phase does not effect the measurement of response phase, regardless of the drive amplitude.
- The check-boxes for `Out 1` and `Out 2` control the output mask. When checked, the drive will be applied at that output port.
- You can select which of the four input ports `In 1 ... In 4` where the MLA will listen for response at any tone.

When you input all of your target values in to the table, select the tuning algorithm that you would like to apply (see *Frequency tuning*):

- `tune 0 (no tuning)` will give you your target values, but this tuning will suffer from Fourier leakage, and it will not be possible to lock in on harmonics or intermodulation products without phase drift.
- `tune 1, prio f, (default)` will tune the measurement bandwidth and frequencies, putting priority on getting as close as possible to the selected frequencies. This is the default tuning scheme that works well for most applications.
- `tune 1, prio df` will tune the measurement bandwidth and frequencies, putting priority on getting as close as possible to the selected measurement bandwidth.
- `tune 2, prio f, (perfect)` will tune the measurement bandwidth and frequencies, putting priority on getting as close as possible to the selected frequencies. In contrast to the `tune 1` options, the `tune 2` options are 'perfect' in that they create exact matches between bandwidth and frequencies. `tune 1` gets very close to a match, but not exact, so that there will be a very small, and for most purposes negligible, amount of Fourier leakage. `tune 1` options also use a periodic correction method to avoid phase drift. The correction is not necessary for `tune 2` options.
- `tune 2, prio df, (perfect)` will tune the measurement bandwidth and frequencies, putting priority on getting as close as possible to the selected measurement bandwidth.

Finally, press the `Write to MLA` button. This will apply the tuning algorithm and configure the MLA™. In order to see what you actually got after the tuning, you need to press `Read from MLA`, and the table will show the values actually used by the MLA™.

With so many tones at your disposal, manual configuration of the MLA™ is obviously impractical when the measurement becomes more complex. In this case you will want to use the Python scripting interface contained in the *Drive Constructor*.

4.5.6 Out of Range Error

The MLA™ is designed to handle signals within a 'normal range' for a 'typical AFM'. Depending on your AFM and our experiment, these normal range settings may need to be adjusted.

When an input signal exceeds the maximum, 'clipping' occurs inside the MLA™ and this induces harmonic and intermodulation distortion in the spectrum. One way to avoid this problem is to simply attenuate the input signal. Intermodulation Products provides attenuators to place at the SIGNAL IN port for this purpose. Note however that a large signal coming from an AFM detector may bring the detector close to saturation, which also gives harmonic and intermodulation distortion of the spectrum. Furthermore, a cantilever will behave with a nonlinear restoring force when it is driven to very large response amplitude. One should be careful not to confuse these sources of nonlinearity, which result in harmonic and intermodulation distortion of the free cantilever oscillation, with harmonic and intermodulation due to a tip-surface force.

Any output signal should also not exceed the maximum or be too small, in order to avoid distortion in the MLA™. Often drive amplitudes are shown in percent of full output. The drive waveform coming from an output port of the MLA™ can be conditioned with external amplifiers or attenuators to suit the application. Internal adjustments can be made inside the MLA™ to give more or less output voltage. Contact Intermodulation Products regarding these adjustments.

4.6 Drive Constructor

The `Drive Constructor` is initially selected in the `Advanced` pull-down menu. The Python scripting interface allows you configure the Multifrequency Lockin Analyzer MLA™. You can use the Drive Constructor to create a frequency comb of drive tones, and to select the frequencies at which you want to perform lockin measurement. Before attempting to use the Drive Constructor, you should read the section on *Intermodulation Measurement*.

4.6.1 Configuring the MLA™

The Python scripting interface has several pre-defined variables and and built-in functions to help in the construction of drive frequency combs. Using the interface requires knowledge of the Python programming language. A pull-down menu lists the name of various Python scripts, stored in **IMP Sessions and Settings/settings/drive_setups/user_drive_scripts.py**. These scripts are well commented to help you understand how they work. You create a new script by changing the name of an old script, and pressing `Save`. Similarly, `Delete` will kill the currently selected script.

The scripting interface has some useful functionality:

- Change font size with ctrl+Plus / ctrl+Minus or ctrl+Mouse wheel.
- Hit ctrl+Space for a list of reserved keywords
- A quick help and link to this manual appears by clicking the IMP help icon .

4.6.1.1 Pre-defined variables

The following variables are defined in all scripts. For some of these variables you will want to change their values, or re-define them, but do not change their shape or data-type. Other variables should not be changed, but you will want to use their values.

- **f0** [real, Hz] and **Q [real]** are the measured resonant frequency and quality factor stored in the *Current calibration*. If you have not run a calibration, these will be zero and your drive script may give an error if you do not set them to a reasonable value in your script.
- **nfreq_out [integer]** and **nfreq_in [integer]** are the number of drive frequencies and response frequencies respectively. These numbers will depend on the MLA™ firmware that you are running. In the most common configurations, **nfreq_out = freq_in**, and in any case, **nfreq_out < nfreq_in**. Do not re-define these values.
- **f_sample [samples per second]** is the sampling frequency of the MLA™. Do not re-define this value.
- **df** [real, Hz], **T** [real, sec], or **samples_per_pixel** [integer]. The measurement bandwidth is set by assigning a value to one (and only one) of these variables. **(df = 1/T = samples_per_pixel/f_sample)**.
- **n [integer array]** or **f [real array, Hz]**. The frequencies are set by assigning values to either of these arrays **(f=n*df)**. The length of these arrays is **nfreq_in** and should not be changed. The first **nfreq_out** entries are the frequencies at which it is possible to drive.
- **a [real array, arbitrary units]**. The drive amplitudes are set by assigning values to this array. The length of **a** is set to be **nfreq_out** and should not be changed. Amplitudes are set relative to each other, in arbitrary units. The maximum or peak amplitude is then set with the amplitude scaling factor (see below).
- **p [real array, Radians]**. The drive phases are set by assigning values to this array. The length of **p** is set to **nfreq_out** and should not be changed. Phases are given in radians.
- **asf [real]** is the amplitude scaling factor, a real number between zero and one. **asf = 1** gives maximum output voltage of the MLA™ at peak amplitude of the waveform. The default value that you get if you do not specify this variable is **asf = 0.1**. An error will appear in the debug window if **asf** is out of the range [0,1]. (Advanced users only: **asf = -1** overrides the scaling of amplitudes, in which case the amplitude values must be given in ADU and care must be taken not to exceed the max output.)
- **out1_mask [Boolean array]** with **nfreq_out** elements. When the element is True (False) it will (will not) send the corresponding tone to port **OUT 1** of the MLA™. The default value of this array is all True.
- **out2_mask [Boolean array]** same as **out1_mask**, but for port **OUT 2** of the MLA™. Note that it is possible to send the same signal to both output ports. The default value of this array is all False.
- **input_multiplex [integer array]** with **nfreq_in** elements. Each element can take on the values 1,2,3 or 4. When an element is 1, the corresponding tone will be measured at port **IN 1**; when the element is 2, the corresponding tone will be measured at port **IN 2**... etc. Note that each tone can be measured at only one input port. The default value is all elements = 1.

Note: The frequency array (**n** or **f**) does not have to be ordered in ascending frequency values. The frequency values can have any order, but there is a correspondence between the first **nfreq_out** elements of the frequency array and the elements of the amplitude array (**a**), phase array (**p**), and the output mask arrays. There is also a correspondence between all elements of the frequency array and input_multiplex array. We access each element of these arrays with the **tone index**. In summary:

- MLA™ tones are specified by the indexed element of an array
- n drive tones [0,1,2,...n]
- m measurement tones [0,1,2,...n,...m]
- first n elements correspond. n<m and usually n=m.

4.6.1.2 Built-in functions

The following built-in functions are provided in all scripts:

- **tune_integer(df)** takes the target value of **df** and returns a re-defined value of **df** that is commensurate with divisions of the MLA™ clock frequency, such that Fourier coefficients will be calculated for perfect integer multiples of **df**. This function is equivalent to the `integer` button in the frequency-tuning controls of *Advanced Setup*. You should use this tuning if you are synchronizing the MLA™ with other instruments.

- **tune_round(df, f0)** takes two arguments, the target value of **df** and a desired frequency commensurate with **df** (e.g. the resonant frequency of the cantilever) . It returns a re-defined value of **df** such that Fourier coefficients will be calculated for near-integer multiples of **df**. This function is equivalent to the `round` button in the frequency-tuning controls of *Advanced Setup*.

4.6.2 Synthesize and Configure

`Synthesize Comb` runs the script and plots the comb as an amplitude-phase plot in the frequency domain, and the waveform in the time domain. The MLA™ can be configured with this comb in one of two ways:

`Configure Drive` causes a voltage signal described by the synthesized frequency comb to be put at the output port of the MLA™.

`Configure Response` adjusts the comb at the output port, such that the synthesized comb appears at the input port. The algorithm that makes this adjustment assumes that a chain of linear systems (e.g. actuator -> cantilever -> detector) is connected between the output and input ports of the MLA™ and it will not work properly when nonlinearity is present in the chain. If the drive voltage to the piezo shaker is not very high and the detector signal is not close to saturation, the AFM is very close to a linear chain. In this case, after pressing Configure Response, the actual cantilever motion have the frequency content given by the comb synthesized from your script.

4.6.3 Setting the feedback

`Setup Feedback` performs the necessary routines to make the free response at the tone index zero, correspond to 100% set-point value. By convention, the MLA™ calculates the feedback error signal from the amplitude of the response at the tone with index zero (frequency specified in the zeroth position of the **n** array or **f** array). You should make sure that your script puts the frequency at which you would like to run feedback in the zero position (see useful code block below). It is probably best to chose the frequency where you get the largest free response for feedback. More information on feedback can be found in the section on *Intermodulation Measurement* (*Feedback*) and in the section on *Advanced Setup* (*Setup Feedback*).

4.6.4 Debugging a script

The `Debug` window allows you to test what is happening with your script. Error messages will appear in this window and you can print any variable here using the **dprint(var_name)** statement in your script. The script automatically prints relevant data for your comb in the Debug window.

4.6.5 Useful Python code blocks

Some useful blocks of code for making frequency combs are described below. These code blocks are used in the example scripts. You can copy and paste these in to your scripts.

Tune **df** close to 500 Hz and set the frequencies to consecutive integer multiples of **df**, centered as close as possible to the resonant frequency **f0**:

```
df = tune_integer(500)
f1 =f0 - (nfreq_out / 2) * df #start freq of comb
n1 = round(f1/df) # integer  closest f1
n = np.int_(arange(nfreq_out) + n1) # integer array of frequencies
```

Tune **df** closer to 500 Hz and set the frequencies to consecutive integer multiples of **df**, centered as close as possible to the resonant frequency **f0**:

```
df = tune_round(500,f0)
f1 =f0 - (nfreq_out / 2) * df #start freq of comb
n1 = round(f1/df) # integer closest f1
n = np.int_(arange(nfreq_out) + n1) # integer array of frequencies
```

Exchange the frequency with the index **mi**, with the feedback frequency at position **n[0]**:

```
#put frequency near resonance as first frequency for feedback
mi=nfreq_out/2 #index of center of comb
n[mi], n[0], a[mi], a[0], p[mi], p[0] = n[0], n[mi], a[0], a[mi], p[0], p[mi]
```

Create an signal in the time domain, consisting of a rapidly oscillating component at the frequency **f_bar** and a slowly modulated amplitude **mod**. Fourier transform and pick out the **nfreq_out** components close to **f_bar** and set the amplitudes *a* and phases *p*:

```
N=2*int(2.*f_bar/df) #number of time samples, so f_max = 2*f_bar
t = arange(0.,1,1./N)/df
#construct signal x(t) in the time domain and FFT to get comb
mod = 0.1*sin(2.*pi*arange(N)/N) # the modulation function
x =(1.+ mod)*cos(2.*pi*f_bar*t) # x(t)
x_hat = fft.rfft(x)
# take components of comb, at frequencies given by n array
a = abs(x_hat[n])
asf = 0.1 # adjust peak amplitude
p = angle(x_hat[n])
```

Create an signal in the time domain, consisting of a rapidly oscillating component with a slowly modulated frequency **f_bar*mod(t)** . Fourier transform and pick out the **nfreq_out** components close to the center frequency **f_bar** and set the amplitudes *a* and phases *p*:

```
N=2*int(2.*n_bar) #number of time samples, so f_max = 2*f_bar
t = (1./df)*arange(0.,1,1./N)
#construct signal x(t) in the time domains
x = cos(2. * pi * ( f_bar * t   -   delta_f /(4*pi*df)*sin(2*pi*df*t)))
# FFT to get frequency comb
x_hat = fft.rfft(x)/(N/2.)
a = abs(x_hat[n])
asf = 0.1 # adjust peak amplitude
p = angle(x_hat[n])
```

4.7 Stream Recorder

When scanning a sample, the MLA™ process end-of-line (EOL) and end-of-file (EOF) triggers and lockin data is stored at pixel locations in a scan file. The stream recorder operates in a different manner: it ignores all triggers and stores data as one continuous, uninterrupted stream. This mode is quite useful for taking data when the AFM probe slowly approachs and retracts from a surface. The data stream can be stored in either *Time mode* or *Frequency mode* (see *Intermodulation Measurement*).

- `Start` will start the stream record. Click again to `Stop`. If the *Autosave* box is checked in the *Status Banner*, each run of the stream recorder will result in a file being saved in the session folder. The file name has the form **record01234** where the number is automatically incremented. In *Time mode* the files will have the **.std** extension, and in *Frequency mode* the files will have the **.imp** extension (see the *File Management* section for more information on file types). Unchecking the `Autosave` box will result in no data being saved.
- `Mode` selects weather the data is stored in *Time mode* or *Frequency mode*. In either mode all data is saved as a continuous stream without any breaks or gaps in the data.

- `Plot` controls only which data is being plotted and does not effect the saving of data. You can adjust the number of points to be plotted in the past number of seconds. The check boxes enable plotting of the amplitudes (or amplitude and phase in the polar plot) at the different frequencies listed.

Note that in the *Time mode* the response at all frequencies in the comb can be examined by analyzing stream data with the Fast Fourier Transform. A script called *PlotTimeMode* (see *Scripting Interface*) is provided which gives a simple GUI inteface for plotting and extracting time-mode data taken with the stream recorder. Below we give an example which can be a useful starting point for your own analysis scripts. The script can be copied to a new filename.py, modified and run, for example in the iPython shell.

4.7.1 Example script

This script reads a time-mode stream data file (.std file) and parses it in to time windows or 'beats', on which the signal is periodic. Each time window is assigned an index (the beat index) and the terminal asks for the beat index on which you want to perform analysis. The script extracts that beat from the data file, and simply plots both time data and the frequency data. Note that you must input the path to the AFM Suite on your computer, and the path to the .std file that you want to analyze:

```
import matplotlib.pyplot as plt
import numpy as np
import sys

# Path to AFM Suite
suitepath = '/Users/david_haviland/IntermodulatorSuite'
sys.path.append(suitepath)
from StreamRecorder import stdfile

# Path to stream recorder file in time-mode
recorder_path = 'path_to_file.std' # full path to recorder file in .std format
print "loading file", recorder_path
sf   = stdfile.stdfile(recorder_path,b_raw=False,do_scale=False)

# Read metadata from file
fs = sf.freq_samples # Sampling frequency, Hz
f1 = sf.f1 # First drive frequency, Hz
f2 = sf.f2 # Second drive frequency, Hz

# Calculate number of samples to avoid Fourier leakage
n_samples_beat = int(np.round(fs/abs(f2-f1))) # Number of samples in one beat
t = 1./fs*np.arange(n_samples_beat) # Time axis for plotting
df = fs/n_samples_beat # Frequency resolution
freqs = df*np.arange(n_samples_beat/2+1) # Frequency axis for plotting

# Index each beat in the stream
ii_s = np.arange(0,sf.n_samples,n_samples_beat)[:-1] # remove last since it might␣
↪not be complete

# Loop through all beats and read max value
beats_to_plot = 1000
if len(ii_s)<=beats_to_plot:
    ii_plot = ii_s
else:
    skip = float(len(ii_s))/beats_to_plot
    ii_plot = np.int_(skip*np.arange(beats_to_plot))

peak = np.zeros(len(ii_plot))
for i,beat_i in enumerate(ii_plot):
    if i%100==0:
        print "Beat %i / %i"%(beat_i, len(ii_s))
    # Read data
    i_start = ii_s[beat_i]
```

(continues on next page)

(continued from previous page)

```
    data = sf.get_samples(i_start,n_samples_beat)
    peak[i] = max(data)

# Plot
plt.figure(1)
plt.clf()
plt.plot(ii_plot,peak)
plt.xlabel("Beat index")
plt.ylabel("Peak value [ADU]")
plt.show()

# Ask terminal for beat number, then data from that beat
n = -1
while n == -1:
    s = raw_input("Enter beat index to plot: ")
    try:
        n = int(s)
    except ValueError:
        print "Input must be an interger"

data = sf.get_samples(ii_s[n],n_samples_beat)
data_fft = np.fft.rfft(data)/len(data)

# Plot
plt.figure(2)
plt.subplot(211)
plt.plot(t,data)
plt.xlabel("Time [s]")
plt.ylabel("In [ADU]")
plt.subplot(212)
plt.semilogy(freqs*1e-3,2*abs(data_fft))
plt.xlabel("Frequency [kHz]")
plt.ylabel("Amplitude [ADU]")
plt.show()
```

4.8 Scripting Interface

The AFM Suite comes with a python scripting interface with which you can develop you own multifrequency AFM methods and customize the IMP Suite to your particular measurement needs. From this interface you can access all the classes and functions used by the IMP Suite for data acquisition, storage, plotting, and analysis. To take full advantage of the possibilities that this interface allows and to develop your own measurements, you will need to learn the Python programing language, and become familiar with the structure and functionality of the IMP Suite.

Intermodulation Products provides a variety of example scripts which can be modified and saved with a new name. All scripts stored in the folder **IMP_sessions_and_settings/settings/scripts** will be loaded when the AFM Suite starts, and they will appear in **Advanced** pull-down menu under **Scripts**. To run the script, simply select it from the menu. `Reload scripts` will close all open scripts are reload the currently saved script.

The file **Examples.py** demonstrates how scripts are constructed as a Python class. This file contains several different classes, each containing a different script with different functionality. These scripts demonstrate how to make a GUI or step a quantity while scanning. You can expand on these scripts to add GUI elements and controls so that parameters can be input and output on the screen. Alterntively, you can simply edit the values in the Python code, reload the script, and run it again. These scripts serve as a starting point to develop your own measurement methods.

Some useful scripts currently available are described below:

4.8.1 RecorderAnalysis.py

`PlotTimeMode` opens a GUI for viewing and extracting sections of a *Stream Recorder* file. Run the script, press `Open` and select a previously recorded **.std** file containing stream data taken in *Time mode*. The script will parse the time stream to a series of 'beats' , or time windows T given by the inverse of the measurement bandwidth $T = 1/\Delta f$, assigning a 'beat index' to each window. A plot shows the maximum value of the oscillation envelope in each beat. You can adjust the time window to include more than one beat, thereby reducing the measurement bandwidth and thus the noise. The time window is shown in the lower plot. `Export All` will export the entire recorder file and `Export zoom` will export only the selected window (lower plot) to an ASCII format file. A Fast Fourier Transform (e.g. FFT) of this data will give the response spectrum at the selected time window.

4.8.2 LiveParampeterMap.py

`LiveParameterMap` runs an experimental new method for extracting polynomial force curves which is computationally very efficient, so that the curves are analyzed and parameter maps are made in real time, as the AFM scans. The method was developed and implemented by Daniel Forchheimer, starting from a theory worked out by Daniel Platz [Platz-2013b]. Choosing this script opens a GUI which allows you to choose ways of parameterizing the conservative tip-surface interaction force. A map of these parameters is created in real time and a force curve showing the extracted parameter can be plotted. The force curve is taken at one location (center) of the fast-scan axis and the plot is updated at the line frequency.

4.9 ScanData

The AFM Software Suite stores data from a scan, including all relevant metadata, in one compact file with the file extension .imp (e.g. scan01234.imp). The **ScanData** class manages the storage and retrival of the data. If you want to write your own Python scripts to analyze multifrequency data taken with the IMP Software Suite, you will need to use the **ScanData** class. Either copy the file from the IMP Software Suite (default path C:\Program Files\IMP Software Suite\Scanner\ScanData.py) and place in the directory of your own script, or import it using the code below.

```
>>> import sys
>>> suite_path = r'C:\Program Files\IMP Software Suite'
>>> sys.path.append(suite_path)
>>> from Scanner import ScanData
```

The documentation below was auto generated from the doc strings in Python code at the time this manual was compiled. You can always see the documentation of the actual class version that you are using in the Python code itself, using auto-complete with any good Python IDE.

4.9.1 ScanData Class

class `afmapp.Scanner.ScanData.`**ScanData**(*filename=None*)

This class handles the storage and retreval of data to and from a scan file (e.g. scan01234.imp). The class is designed to be easily extendable and new functions will be added as the IMP Suite grows. Most functions set data in, or get data from the containers provided in the class.

Units are specified in square brackets in the doc strings. In additon to SI units (e.g. [Hz], [sec], [Volts], ...) some functions use the following native lockin, or digital units:

- PCU - phase counter units, an integer number between 0 and 2**42.
- ADU - analog to digital units, integer steps of an AD converter.
- DAU - digital to analog units, integer steps of a DA converter

Examples

```
>>> # instanciate class and load data from file
>>> sd = ScanData.ScanData('scan01612.imp')
>>> # get the frequency array, units Hz
>>> freqs = sd.get_karray()*sd.get_df()
>>> # load all image data in to a large array, complex format, untis [ADU]
>>> image_data = sd.get_data()
>>> # pick out the spectrm at one pixel x=100, y=99, and convert to units [nm]
>>> pixel = image_data[:, 99, 100]*sd.get_invOLR()
>>> # convert from complex format to amplitude and phase
>>> amps = np.abs(pixel)
>>> phases = np.arctan2( np.imag(pixel), np.real(pixel) )
```

Note: Functions in this class do not set or get the actual measurement conditions of the multi-frequency lockin. Setting data read from a file does not change the file on disk, unless the data is written again with `ScanData.ScanData.save_to_file()`

---

get_Q(*mode_type='flexural'*, *mode_nr=1*)
Get the cantilever quality factor.

Parameters

- **mode_type** (*str, optional*) – Name of mode, i.e. 'flexural' or 'torsional'
- **mode_nr** (*int, optional*) – Mode number, i.e. 1, 2, 3, ...

Returns mode quality factor

Return type float

Raises `CalibrationError` – if the mode was not calibrated

get_amplitudes(*unit='digital'*)
Drive amplitudes for every tone. Undriven tones have zero amplitude.

Parameters **unit** (*str, optional*) – {'digital', 'ADU', 'calibrated', 'V', 'percent', '%'}

Raises

- `CalibrationError` – if the specified units require conversion, but no MLA calibration was set
- `ValueError` – if unknown unit is specified

get_calib_mode_index(*mode_type=None*, *mode_nr=1*)
Find the index of calibration (if exist) in self.calib_list. Else return False and index=None.

Parameters

- **mode_type** (*str, optional*) – type of cantilever mode, flexural or torsional
- **mode_nr** (*int, optional*) – mode number

Returns

cal_exists, mode_index bool: True if the calibration was found, False otherwise int: the index of the required calibration. None if not found.

Return type tuple

get_cantilever_noise(*mode_type='flexural'*, *mode_nr=1*)
Get the thermal noise force.

Parameters

- **mode_type** (*str, optional*) – Name of mode, i.e. 'flexural' or 'torsional'

- **mode_nr** (*int, optional*) – Mode number, i.e. 1, 2, 3, ...

Returns cantilever thermal noise force in ADU**2 / Hz

Return type float

Raises CalibrationError – if the mode was not calibrated

Examples

```
>>> sd = ScanData.ScanData('scan01612.imp')
>>> psd = np.sqrt(sd.get_cantilever_noise()) * sd.get_k() * sd.get_
↪invOLR() * 1e15
>>> print "Cantilever thermal noise force is {:.2f} fN/sqrt(Hz)".
↪format(psd)
Cantilever thermal noise force is 21.90 fN/sqrt(Hz)
```

get_data(*data_slice=(slice(None, None, None), slice(None, None, None), slice(None, None, None))*, *scan_pass='trace'*, *data_format='complex'*, *unit='ADU'*)

The data measured during the scan.

Parameters

- **data_slice** (*tuple, optional*) – tuple of Python slice objects (tone_slice, y_slice, x_slice).
- **scan_pass** (*str or int, optional*) – {'trace', 'retrace', 'interleave trace', 'interleave retrace'} specifies scan pass.
- **data_format** (*str, optional*) – {'IQreal', 'complex', 'amp', 'phase'} specifies data type.
- **unit** (*str, optional*) – {'ADU', 'V', 'm', 'nm', 'rad', 'deg'} specifies units. 'rad' and 'deg' only for phase.

Returns slice of the 3D data set. The data set has shape [index of tone, pixel y coord , pixel x coord]

Return type np.ndarray

Examples

```
>>> sd = ScanData.ScanData('scan01612.imp')
>>> # get engaged response at pixel x=99 y=88, trace, complex values, in␣
↪nm.
>>> data = sd.get_data(unit='nm')
>>> pixel = data[ : , 88 , 99 ]
>>> # alternative way, which avoids large object 'data'
>>> data_slice = np.s_[ : , 88 , 99 ]
>>> pixel = sd.get_data( data_slice, unit='nm')
>>> # get phase image at tone number 15, retrace, in degrees.
>>> data_slice = np.s_[ 15 , : , : ]
>>> phase_image = sd.get_data( data_slice, scan_pass='retrace', data_
↪format='phase', unit='deg')
```

Raises

- CalibrationError – if the specified units require conversion, but no MLA calibration was set
- ValueError – if the specified format and units are incompatible, or the scan pass, format or units are unknown

get_detector_noise(*mode_type='flexural'*, *mode_nr=1*)
Get the detector noise floor.

Parameters

- **mode_type** (*str, optional*) – Name of mode, i.e. 'flexural' or 'torsional'
- **mode_nr** (*int, optional*) – Mode number, i.e. 1, 2, 3, ...

Returns detector noise floor in ADU**2 / Hz

Return type float

Raises CalibrationError – if the mode was not calibrated

Examples

```
>>> sd = ScanData.ScanData('scan01612.imp')
>>> psd = np.sqrt(sd.get_detector_noise()) * sd.get_invOLR() * 1e15
>>> print "The detector noise floor is {:.2f} fm/sqrt(Hz)".format(psd)
The detector noise floor is 138.13 fm/sqrt(Hz)
>>> psd = np.sqrt(sd.get_detector_noise()) * sd.get_invOLR() * sd.get_k()␣
↪* 1e15
>>> print "The equivalent noise force is {:.0f} fN/sqrt(Hz)".format(psd)
The equivalent noise force is 5013 fN/sqrt(Hz)
```

get_df()
Measurement bandwidth.

Returns Hertz, Hz

Return type float

Examples

```
>>> sd = ScanData.ScanData('scan01612.imp')
>>> df = sd.get_df()
```

get_drive_amplitudes()
Amplitudes of driven tones.

Returns lockin output amplitude digital units (LOA)

Return type np.ndarray(np.int64)

See also:

get_drive_karray() get_drive_phases()

get_drive_karray()
Frequencies of driven tones, as multiples of measurement bandwidth (df).

Returns frequencies where drive amplitudes are non-zero

Return type np.ndarray(dtype=np.int64)

Examples

```
>>> sd = ScanData.ScanData('scan01612.imp')
>>> # create frequency array for drive frequencies, in Hz
>>> freqs = sd.get_drive_karray() * sd.get_df()
```

See also:

get_drive_amplitudes() get_drive_phases()

get_drive_phases()
Phases of driven tones.

Returns radians

Return type np.ndarray(dtype=np.float64)

See also:

get_drive_karray() get_drive_amplitudes()

get_endtime()
Time at end of scan, in seconds since the epoch.

Returns seconds

Return type float

get_external_filename()
The filename of an external file associated with this scan.

Returns filename

Return type str

Note: Currently not implimented.

---

get_f0(*mode_type='flexural'*, *mode_nr=1*)
Get the cantilever resonance frequency.

Parameters

- **mode_type** (*str, optional*) – Name of mode, i.e. 'flexural' or 'torsional'
- **mode_nr** (*int, optional*) – Mode number, i.e. 1, 2, 3, ...

Returns mode resonance frequency in Hz

Return type float

Raises CalibrationError – if the mode was not calibrated

get_feedback()
Feedback gains of the host AFM.

Returns [i_gain, p_gain] of host AFM. Will return zero if not implimented on host AFM.

Return type np.ndarray(dtype=float64)

get_free(*data_format='complex'*, *unit='ADU'*)
Spectrum of free cantilever oscillation.

Parameters

- **data_format** (*str, optional*) – {'IQreal', 'complex', 'amp', 'phase'} specifies data type.
- **unit** (*str, optional*) – {'ADU', 'V', 'm', 'nm', 'rad', 'deg'} specifies units. 'rad' and 'deg' only for phase.

Returns spectrum of free cantilever oscillation, with required data format and units

Return type np.ndarray

Raises

- CalibrationError – if the specified units require conversion, but no MLA calibration was set

- `ValueError` – if the specified format and units are incompatible, or the scan pass, format or units are unknown

See also:

get_lift()

Examples

```
>>> sd = ScanData.ScanData('scan01612.imp')
>>> # get free response in nm.
>>> data = sd.get_free(unit='nm')
>>> freqs = sd.get_karray()*sd.get_df()
>>> #plot amplitude vs. frequency
>>> l, = plot(freqs, np.abs(data))
```

get_frequencies()
Frequencies of the measured tones.

Returns Hertz, Hz

Return type np.ndarray(dtype=np.float64

get_fs()
Lockin sampling frequency.

Returns samples/second, Hz

Return type float

get_imp_str_list(*mode=-1*)
Function to create a string list to lable the tones

Parameters **mode** (*int, optional*) – {-1,0,1,2,3}

Returns strings with names of imp's, e.g. 'Drive1', 'Drive2', 'IMP3L', etc.

Return type list

mode	Returns
-1	strings as saved in ImageData
0	as mode=-1, but empty strings, automatic mode==2
1	as mode=-1, but empty strings, automatic mode==3
2	strings of karray_imp integers e.g 'k = 345'
3	strings as defined in ScannerPannel e.g.

To Do: depreciate this function and have imp_str_list set by user in drive constructor or set to default value in auto setup.

get_input_ports()
Input port for each of the measured tones.

Returns array of port numbers [1-4] where tones are measured

Return type np.ndarray(dtype=np.int64)

See also:

`mlaapi.lockin.Lockin.get_port()` `mlaapi.lockin.Lockin.set_input_multiplexer()`

get_invOLR(*mode_type='flexural'*, *mode_nr=1*)
Get the inverse optical lever responsivity.

Parameters

- **mode_type** (*str, optional*) – Name of mode, i.e. 'flexural' or 'torsional'
- **mode_nr** (*int, optional*) – Mode number, i.e. 1, 2, 3, ...

Returns mode invOLR in m/ADU

Return type float

Raises CalibrationError – if the mode was not calibrated

get_k(*mode_type='flexural'*, *mode_nr=1*)
Get the cantilever stiffness.

Parameters

- **mode_type** (*str, optional*) – Name of mode, i.e. 'flexural' or 'torsional'
- **mode_nr** (*int, optional*) – Mode number, i.e. 1, 2, 3, ...

Returns mode stiffness in N/m

Return type float

Raises CalibrationError – if the mode was not calibrated

get_karray()
Frequencies of measured tones, as multiples of measurement bandwidth (df).

Returns

Return type np.ndarray(dtype=np.int64)

Examples

```
>>> sd = ScanData.ScanData('scan01612.imp')
>>> # create array of frequencies in [Hz] where response was measured
>>> freqs = sd.get_karray() * sd.get_df()
```

get_lift(*data_format='complex'*, *unit='ADU'*)
Spectrum of cantilever oscillation at the lifted scanner position.

Parameters

- **data_format** (*str, optional*) – {'IQreal', 'complex', 'amp', 'phase'} specifies data type.
- **unit** (*str, optional*) – {'ADU', 'V', 'm', 'nm', 'rad', 'deg'} specifies units. 'rad' and 'deg' only for phase.

Returns spectrum of lifted cantilever oscillation, with required data format and units

Return type np.ndarray

Raises

- CalibrationError – if the specified units require conversion, but no MLA calibration was set
- ValueError – if the specified format and units are incompatible, or the scan pass, format or units are unknown

---

Note: Used for separating background forces from nonlinear forces

---

See also:

get_free()

get_loaded_filename()
The filename used when loading ScanData from disk.

Returns filename

Return type str

get_metadata_ascii()
Return all metadata as a string new-line separated string

Returns str

Strategy for all following data entries: IDENTIFIER[UNIT]: DATA

get_mla_calibration()
The hardware calibration of the MLA.

Returns dict

get_mla_firmware_version()
Version of the firmware loaded into the MLA.

Returns version number

Return type int

See also:

`mlaapi.hardware.Hardware.get_firmware_version()`

get_original_filename()
The file name when the data was first stored.

Returns filename

Return type str

get_output_mask()
Each bit in the mask determines weather the corresponing tone should be added to the output.

Returns a list of int with the mask for each of the two output ports

Return type list

See also:

`mlaapi.lockin.Lockin.set_output_mask()` `mlaapi.utils.mask_to_int()`

get_phases(*unit='rad'*)
Drive phases for every tone.

Parameters **unit** (*str, optional*) – {'rad', 'deg'}

Raises `ValueError` – if unknown unit is specified

get_resolution()
Get the scan resolution.

Returns

(nx, ny) int: number of pixels in fast-scan direction. int: number of pixels in slow-scan direction.

Return type tuple

get_samples_per_pixel()
Number of samples in time window corresponding to one image pixel, fs/df.

Returns samples per pixel

Return type float

get_scan_rate()
Number of scan lines per second (trace + retrace).

Returns Hertz, Hz

Return type float

get_scan_size_x()

Returns scan size in the fast-scan direction x in meters

Return type float

Note: Scan size is determined by the host AFM. It is either entered manually in the GUI, or automaically when reading in height data. Automatic over-writes manual.

---

get_scan_size_y()

Returns scan size in the slow-scan direction y in meters

Return type float

Note: Scan size is determined by the host AFM. It is either entered manually in the GUI, or automaically when reading in height data. Automatic over-writes manual.

---

get_scan_subtype()
Additional scan sub-type identifier.

Returns str

get_scan_type()
Scan type identifier (e.g. ImAFM, ImEFM, ImFFM, . . .).

Returns str

get_setpoint()
Amplitude setpoint.

Returns

value of setpoint (at end of scan). Value between 0 and 1 is the fraction of free oscillation amplitude at the feedback tone with index 0n (i.e. first element of frequency array and response arrays)

Return type float

get_starttime()
Time at start of scan, in seconds since the epoch.

Returns seconds

Return type float

get_tip_Vac()
AC voltage applied to the tip in ImEFM.

Returns Volts, V

Return type float

get_tip_Vdc()
DC voltage applied to the tip in ImEFM.

Returns Volts, V

Return type float

given_list_is_empty(*lst*)

Returns True if list is [] or only has elements of type ‘’, otherwise False.

Return type bool

has_calibration(*mode_type='flexural'*, *mode_nr=1*)
Check if a valid calibration is stored.

Parameters

- **mode_type** (*str, optional*) – type of cantilever mode, flexural or torsional
- **mode_nr** (*int, optional*) – mode number

Returns True for a valid calibration

Return type bool

load_from_file(*filename*, *minimal=False*)
Summary

Parameters

- **filename** (*str*) – filename to be opened
- **minimal** (*bool, optional*) – if True, only load parameters

Raises FileVersionError – if file version not supported

reset_lift()
Empty previously stored lift data.

save_to_ascii(*dirname=None*)
Export to a collection of generically named ascii-files in the directory ‘dirname’.

Creates a separate subdirectory for the scan and separate files for each amplitude and phase image for easy import into Matlab or similar.

Parameters **dirname** (*str, optional*) – if None, the directory is automatically named after the ScanData file

save_to_file(*filename*)
Saves all data in the ScanData class to an hdf5 file.

Parameters **filename** (*str*) – desired filename

set_Q(*Q*, *mode_type='flexural'*, *mode_nr=1*, *verbose=True*)
Set the cantilever quality factor for an existing calibration.

Parameters

- **Q** (*float*) – quality factor
- **mode_type** (*str, optional*) – Name of mode, i.e. ‘flexural’ or ‘torsional’
- **mode_nr** (*int, optional*) – Mode number, i.e. 1, 2, 3, ...
- **verbose** (*bool, optional*) – If True, print a message to the console

Raises CalibrationError – if the mode was not calibrated

set_amplitudes(*amps_digital*)
Drive amplitudes for every tone. Undriven tones have zero amplitude.

Parameters **amps_digital** (*np.ndarray(dtype=np.int64)*) – digital units, ADU

set_calibration(*mode_type*, *mode_nr*, *file_name*, *f0*, *Q*, *k*, *invOLR*, *cantilever_noise*, *detector_noise*)
Set calibration based on mode_type and mode_nr.

Parameters

- **mode_type** (*str*) – Name of mode, i.e. ‘flexural’ or ‘torsional’

- **mode_nr** (*int*) – Mode number, i.e. 1, 2, 3, ...
- **file_name** (*str*) – Name of calibration file belonging to *this* calibration
- **f0** (*float*) – Mode resonance frequency in Hz
- **Q** (*float*) – Mode quality factor
- **k** (*float*) – Mode stiffness in N/m
- **invOLR** (*float*) – Inverse optical lever responsitivy in m/ADU
- **cantilever_noise** (*float*) – Cantilever noise in ADU**2/Hz
- **detector_noise** (*float*) – Detector noise in ADU**2/Hz

Note: If mode is already calibrated, overwrite current calibration. If it doesn't exist, add it to self.calib_list.

---

set_cantilever_noise (*p_thermal*, *mode_type='flexural'*, *mode_nr=1*, *verbose=True*)
Set the thermal noise force for an existing calibration.

Parameters

- **p_thermal** (*float*) – thermal noise force in ADU**2 / Hz
- **mode_type** (*str, optional*) – Name of mode, i.e. 'flexural' or 'torsional'
- **mode_nr** (*int, optional*) – Mode number, i.e. 1, 2, 3, ...
- **verbose** (*bool, optional*) – If True, print a message to the console

Raises CalibrationError – if the mode was not calibrated

set_detector_noise (*p_detector_white*, *mode_type='flexural'*, *mode_nr=1*, *verbose=True*)
Set the detector noise floor for an existing calibration.

Parameters

- **p_thermal** (*float*) – thermal noise force in ADU**2 / Hz
- **mode_type** (*str, optional*) – Name of mode, i.e. 'flexural' or 'torsional'
- **mode_nr** (*int, optional*) – Mode number, i.e. 1, 2, 3, ...
- **verbose** (*bool, optional*) – If True, print a message to the console

Raises CalibrationError – if the mode was not calibrated

set_df (*df*)
Measurement bandwidth.

Parameters **df** (*float*) – Hertz, Hz

See also:

set_samples_per_pixel()

set_endtime (*t0*)
Time at end of scan, in seconds since the epoch.

Parameters **t0** (*float*) – seconds

set_external_filename (*filename*)
The filename of an external file associated with this scan.

Parameters **filename** (*str*) – filename

Note: Currently not implimented.

---

set_f0 (*f0*, *mode_type='flexural'*, *mode_nr=1*, *verbose=True*)
Set the cantilever resonance frequency for an existing calibration.

Parameters

- **f0** (*float*) – resonance frequency in Hz
- **mode_type** (*str, optional*) – Name of mode, i.e. 'flexural' or 'torsional'
- **mode_nr** (*int, optional*) – Mode number, i.e. 1, 2, 3, ...
- **verbose** (*bool, optional*) – If True, print a message to the console

Raises CalibrationError – if the mode was not calibrated

set_feedback (*array*)
Feedback gains of the host AFM.

Parameters **np.ndarray** (*dtype=float64*) – feedback parameters of host AFM [i_gain, p_gain]. Not implimented on all host AFMs.

set_free (*free*, *data_format='complex'*, *unit='ADU'*)
Spectrum of free cantilever oscillation.

Parameters

- **free** (*np.ndarray*) – spectrum of free cantilever oscillation with specified data format and units
- **data_format** (*str, optional*) – {'IQreal', 'complex'} specifies data type.
- **unit** (*str, optional*) – {'ADU', 'V', 'm', 'nm', 'rad', 'deg'} specifies units. 'rad' and 'deg' only for phase.

Raises

- CalibrationError – if the specified units require conversion, but no MLA calibration was set
- ValueError – if the specified format and units are incompatible, or the scan pass, format or units are unknown

See also:

set_lift()

Examples

```
>>> sd = ScanData.ScanData('scan01612.imp')
>>> # get free response in nm.
>>> data = sd.get_free(unit='nm')
>>> freqs = sd.get_karray()*sd.get_df()
>>> #plot amplitude vs. frequency
>>> l, = plot(freqs, np.abs(data))
```

set_frequencies (*freqs_Hz*)
Frequencies of the measured tones.

Parameters **freqs_Hz** (*np.ndarray (dtype=np.float64)*) – Hertz, Hz

set_fs (*fs*)
Lockin sampling frequency.

Parameters **fs** (*float*) – samples/second, Hz

set_input_ports (*ports*)
Input port for each of the measured tones.

Parameters **np.ndarray** (*dtype=np.int64*) – array of port numbers [1-4] where tones are measured

See also:

`mlaapi.lockin.Lockin.get_port()` `mlaapi.lockin.Lockin.set_input_multiplexer()`

set_invOLR(*invOLR*, *mode_type='flexural'*, *mode_nr=1*, *verbose=True*)
Set the inverse optical lever reposponsivity for an existing calibration.

Parameters

- **invOLR** (*float*) – inverse optical lever reposponsivity in m/ADU
- **mode_type** (*str, optional*) – Name of mode, i.e. 'flexural' or 'torsional'
- **mode_nr** (*int, optional*) – Mode number, i.e. 1, 2, 3, ...
- **verbose** (*bool, optional*) – If True, print a message to the console

Raises `CalibrationError` – if the mode was not calibrated

set_k(*k*, *mode_type='flexural'*, *mode_nr=1*, *verbose=True*)
Set the cantilever stiffness for an existing calibration.

Parameters

- **k** (*float*) – stiffness in N/m
- **mode_type** (*str, optional*) – Name of mode, i.e. 'flexural' or 'torsional'
- **mode_nr** (*int, optional*) – Mode number, i.e. 1, 2, 3, ...
- **verbose** (*bool, optional*) – If True, print a message to the console

Raises `CalibrationError` – if the mode was not calibrated

set_lift(*lift*, *data_format='complex'*, *unit='ADU'*)
Spectrum of cantilever oscillation at the lifted scanner position.

Parameters

- **lift** (*np.ndarray*) – spectrum of lift cantilever oscillation with specified data format and units
- **data_format** (*str, optional*) – {'IQreal', 'complex'} specifies data type.
- **unit** (*str, optional*) – {'ADU', 'V', 'm', 'nm'} specifies units.

Raises

- `CalibrationError` – if the specified units require conversion, but no MLA calibration was set
- `ValueError` – if the specified format and units are incompatible, or the scan pass, format or units are unknown

---

Note: Used for separating background forces from nonlinear forces

---

See also:

set_free()

set_mla_all_settings(*mla*)
Convenience function to set all MLA related settings from a mla instance.

Parameters **mla** (*mlaapi.mla_suite.MLA*) – object instance of MLA

Note: This method will set *self.nr_input_freq*, *self.nr_output_freq* and call the methods listed in "See also"

See also:

set_fs() *set_mla_calibration()* *set_mla_firmware_version()* *set_samples_per_pixel()* *set_output_mask()* *set_input_ports()* *set_amplitudes()* *set_phases()* *set_frequencies()*

set_mla_calibration(*cal*)
The hardware calibration of the MLA.

Parameters **cal** (*dict*) –

set_mla_firmware_version(*ver*)
Version of the firmware loaded into the MLA.

Parameters **ver** (*int*) – version number

See also:

`mlaapi.hardware.Hardware.get_firmware_version()`

set_output_mask(*mask*)
Each bit in the mask determines weather the corresponing tone should be added to the output.

Parameters **mask** (*list*) – a list of int with the mask for each of the two output ports

See also:

`mlaapi.lockin.Lockin.set_output_mask()` `mlaapi.utils.mask_to_int()`

set_phases(*phases_rad*)
Drive phases for every tone.

Parameters **phases_rad** (*np.ndarray(dtype=np.float64)*) – radians, rad

set_resolution(*nx*, *ny*)
Set the scan resolution.

Parameters

- **nx** (*int*) – number of pixels in fast-scan direction.
- **ny** (*int*) – number of pixels in slow-scan direction.

set_samples_per_pixel(*samples_per_pixel*)
Number of samples in time window corresponding to one image pixel, fs/df.

Parameters **samples_per_pixel** (*float*) – samples per pixel

set_scan_rate(*scan_rate*)
Number of scan lines per second (trace + retrace).

Parameters **scan_rate** (*float*) – Hertz, Hz

set_scan_size_x(*value*)

Parameters **value** (*float*) – scan size in the fast-scan direction x in meters

set_scan_size_y(*value*)

Parameters **value** (*float*) – scan size in the slow-scan diretion y in meters

set_scan_subtype(*scan_subtype*)
Additional scan sub-type identifier.

Parameters **scan_subtype** (*str*) –

set_scan_type(*scan_type*)
Scan type identifier (e.g. ImAFM, ImEFM, ImFFM, . . .).

Parameters **scan_type** (*str*) –

set_setpoint (*value*)
Amplitude setpoint.

Parameters **value** (*float*) – between 0 and 1 is the fraction of free oscillation amplitude at the feedback tone with index 0.

set_starttime (*t0*)
Time at start of scan, in seconds since the epoch.

Parameters **t0** (*float*) – seconds

set_tip_Vac (*value*)
AC voltage applied to the tip in ImEFM.

Parameters **value** (*float*) – Volts, V

set_tip_Vdc (*value*)
DC voltage applied to the tip in ImEFM.

Parameters **value** (*float*) – Volts, V

4.10 File Management

The IMP suite is kept in different places, depending on your computer system. In Windows, it is installed in **c://Program Files/IntermodulatorSuite**. Users do not need to care about this folder as it contains only static files which should be installed from an installer. All dynamic files that are changed when you change settings in the AFM Suite, store data, save a script, etc., are stored in the folder **c://IMP sessions and settings**. This folder has the following sub-folders:

Moving data to another computer

To transport your data to another computer for analysis, be sure to first import the host AFM data (i.e. height image data) in to your session folder using the *Session Overview*. Then simply copy the entire session folder to another computer with the AFM Suite installed. When you perform analysis on another computer, all newly created analysis files will be added to the session folder.

/sessions This folder contains session folders. Each time you start the AFM Suite a dialog box opens to create a new, or add to an existing session folder. The default name of the new session folder is the date, but you have the option to append a session name to the date. When Autosave is checked (see *Status Banner*), data is automatically stored in the session folder: scan files, calibration files, stream files, the session log, etc.. When you analyze data to create new files or import host AFM data files, the default behavior is to always store everything in the session folder.

/settings This folder contains many files that are changed dynamically by the AFM Suite and configuration files for different AFMs. Normal users do not need to work directly with these files but advanced users may want to modify them. If these files become corrupt, they can be discarded and the IMP Suite will automatically recreate the files with their default values on the next start of the IMP Suite. The settings folder has several sub-folders:

/calibration Contains several text files with the calibration constants for each type of cantilever programmed in the AFM Suite, including those created by a user. There is also a file **fluid.txt** containing the density and viscosity of the various fluids that you may want to work in.

/configurations Contains files for configuring the AFM Suite to work with your particular host AFM. The normal user will not need to be concerned with these files.

/drive_setups Contains the file **user_drive_setups.py** which is a Python file containing the scripts for the *Drive Constructor*, both example scripts and user-constructed scripts.

/force_models Contains Python files with the force models. You can add your own force model files here by copying and modifying **/example.py**. When you restart the AFM Suite this file will be re-loaded and your model will appear as a new force model.

/scripts Contains Python scripts which can be sued to control different measurement schemes and create different modes. This is for advanced users who will require some knowledge of the basic Python objects in the AFM Suite. A few examples are given in **/Examples.py**

/config.ini This is the main configuration file which is dynamically changed by the AFM Suite. The normal user will not need to be concerned with this file. If it becomes corrupted, it can be deleted and a new file will be created with default values upon restart of the IMP Suite.

If you run the MLA™ as a stand-alone instruent, independant of the AFM Software Suite for ImAFM™, there will also be other files and folders in the main **IMP sessions and settings** folder. These files and folders are described in a seperate manual dedicated to the MLA™ alone.

4.10.1 File Types

The AFM Suite generates files with a name indicating the type of file, followed by an automatically incremented 5-digit counting number (denoted '01234' below). You are free to append an underscore followed by any valid file text **after** this counting number (e.g. scan01234_my-file-text.imp). You may experience problems keeping track of your files if you remove or modify this number or the type name before it. The following files are generated by the IMP Suite and stored in a session folder:

- **calib01234.txt** noise data and fitted calibration parameters (see *Calibration*).
- **calib01234.png** image of plot in calibration view.
- **sweep01234.txt** frequency sweep data (see *Frequency Sweep*).
- **sweep01234.png** image of the frequency sweep plot.
- **scan01234.imp** data for one scan, including the current calibration, in HDF5 format.
- **recorder01234.imp** continuous stream of data taken in frequency mode (see *Stream Recorder*).
- **recorder01234.std** continuous stream of data taken in time mode.
- **scan01234_model.npz** parameter map data for the fitted **model** (see *Parameter Maps*).
- **scan01234_p-name.png** color map image of the parameter **p-name**.
- **sessionlog.txt** chronological session log (see *Session Logbook*).

You can always export an image of any plot that you see on screen by clicking the save image icon in the *Image Toolbar*, which will bring up a save dialog box with several file format options. You can also export raw data and the results of analysis to a text file using the *Data Tree* .

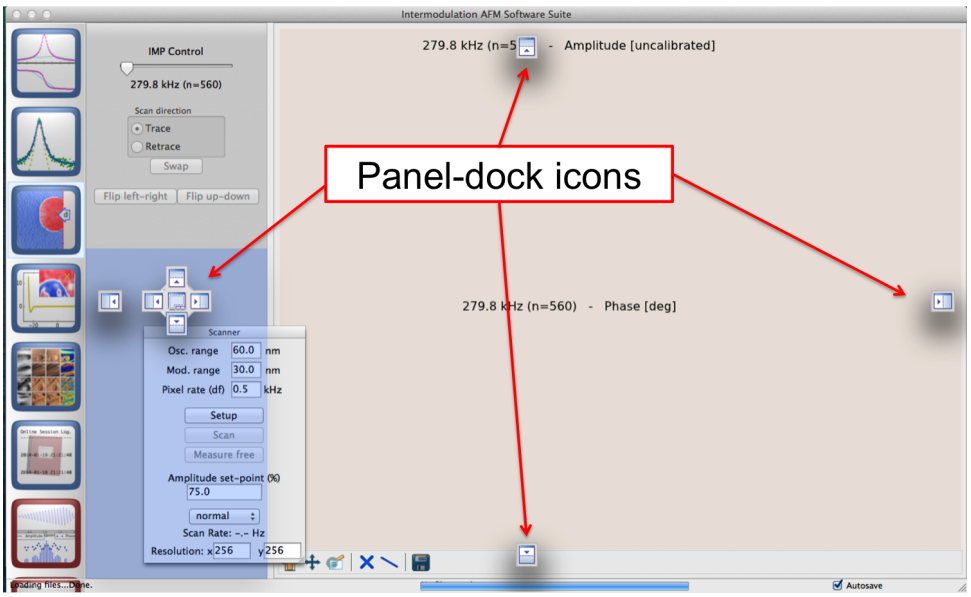
4.11 Panels and Views

All views, or screen layouts, have a main panel to which auxiliary panels can be docked. Several views are pre-defined and they appear when clicking icons in the left column symbolizing the *Work Flow*. Icons for the advanced panels appear when selected in the `advanced` pull-down menu. To removed the advanced icons, select again in the pull-down menu.

Different panels appear when you click `Settings` buttons, or they can be chosen from the `Analysis` pull-down menu. You can add these panels to the views, or rearrange the panels in a view according to your desire. Movable panels have a shaded grab-bar on top, where the panel name is located. These panels can be free-floating, or you can dock them to a particular view.

- To un-dock a panel, click-and-drag the panel bar. Release when it is un-docked and it will be free-floating. A double-click on the panel bar will also un-dock the panel.

- To dock a panel, click-and-drag the panel bar of a free-floating panel and panel position icons will appear. Move your mouse cursor to the desired dock location on the icon, and your screen will show a shaded area. Upon release the panel will dock to the shaded area.



CHAPTER

FIVE

INSTALLATION

5.1 Install Software

The AFM Software Suite is installed with a standard installer. When you install the software, you agree to the terms of the IMP license. The IMP software Suite can be run on the same computer as the host AFM, or on a separate computer. It is advantageous to run the AFM Suite on a modern computer with a multi-core processor, especially when analyzing full images to make *Parameter Maps*.

If the AFM Suite does not detect the The Multifrequency Lockin Analyzer™ (MLA™) upon starting, it will open up in analysis mode. In analysis mode you can plot and analyze data taken previously, but you can not take data. You are free to install the software on as many seperate computers needed to analyze your data. To take data you mast start the IMP Suite when connected to the MLA™ and a host AFM, as described below.

5.2 Install Hardware

Connecting the hardware the first time requires some effort. However, if things are configured nicely, one or two hardware switches is all that is needed to switch between the normal configuration of your host AFM and the ImAFM™ configuration. The MLA™ is connected to the host AFM via a signal access module (SAM) provide by Intermodulation Products or your AFM supplier. The MLA™ is connected to a computer running the AFM Suite via Ethernet, either the same computer used by the host AFM, or a seperate computer.

5.2.1 Connection to computer

The AFM Suite will open analysis mode when it does not detect an MLA™ connected to the computer. To establish the connection between the computer and the MLA™ you must first perform the steps below. After this procedure the detection will be automatic. The instructions given below are for **Windows 7**.

1. Connect a standard Ethernet cable between the MLA™ and the computer.
2. Go to `Control Panel` -> `Network and Internet` -> `Network and Sharing Center`.
3. **Locate the network adapter:** In the left panel choose `Change adapter settings`. To identify which adapter corresponds to the Ethernet card connected to the MLA™, turn off the MLA™. There should be a red cross on the adapter icon with the text `Network cable unplugged`. About one minute after the MLA™ is turned on the red cross will disappear and the text changes to `Unidentified network`.
4. **Configure the MLA adapter:** Right click again on the MLA adapter and choose `Properties`. Select `Internet Protocol Version 4 (TCP/IPV4)` and click `Properties`. Click `Use the following IP address` and fill in the following

   IP address: 192.168.42.1

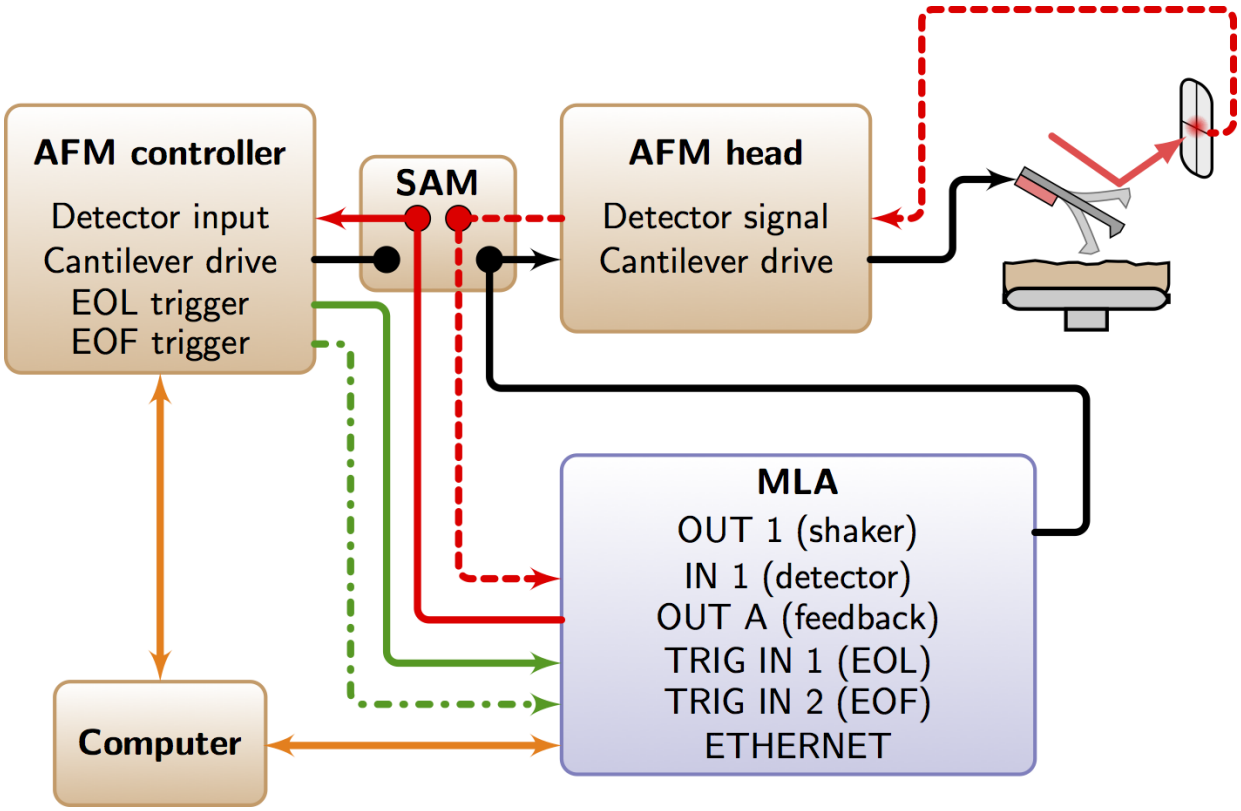
   Subnet mask: 255.255.255.0

5. **Check the adapter:** At this point the internet connection between the MLA™ and the computer should be established. To test the connection you can ping the MLA™. In a Windows terminal window type `ping 192.168.42.50`. The MLA™ should respond with something like:

Pinging 192.168.42.50 with 32 bytes of data:

Reply from 192.168.42.50: bytes=32 time=1ms TTL=128

5.2.2 Connection to host AFM



The general connection scheme is shown the diagram above. The cantilever drive, detector, feedback and trigger signals must be routed through the MLA™. This routuing may be with with BNC cables to the font pannel of the MLA™, or with a composite cable to the **AUX** connection of the back panel of the MLA™. On some AFMs a separate Signal Access Module (SAM) is required. Intermodulation Products provides dedicated SAMs for a few common AFMs to easily toggle between the normal function of your host AFM and ImAFM™. On other AFMs, notably Asylum and JPK, one connects BNC cables directly to the controller and changes signal routing using software activated switches. You can get a script from Intermodulation Products to automatically perform the signal routing. Detailed instructions are given below for different AFM's. Contact **Intermodulation Products** to find out what solutions exist for your particular AFM <info@intermodulation-products.com>.

5.3 Set AFM Type

In the AFM Suite, under the `Advanced` pull-down menu, select `Setup AFM`. A dialog box appears with a pull-down menu to choose your AFM type. This option will only appear when the IMP Suite detects the MLA™ and starts in full mode, ready for taking data. The settings for the various AFM types are stored in the folder **IMP Sessions and Settings/settings/configurations**. These files contain information about triggers, over-scan factors, and more. Advanced users can use the files in this folder as a template to make their own **custom_afm.ini** settings file. When the software is restarted, a new **custom_afm** settings will appear as a choice in the pull-down menu.

If your AFM type does not have a configuration file, contact **Intermodulation Products** for help in making a custom configuration file <info@intermodulation-products.com>.

5.3.1 Working without triggers

Nearly all AFM's have some kind of access to the end-of-frame (EOF) and end-of-line (EOL) triggers. If the EOF trigger does not exist, it is still possible to manually start the ImAFM™ scan when the host AFM scan starts. However, after several scans a manual restart is needed to re-sync the frames. If the EOL trigger does not exist, it is also possible perform ImAFM™ but in this case one needs to accurately adjust the timing in order to keep ImAFM™ in sync with the host AFM.

Under the `Advanced` pull-down menu, select `Setup AFM`. Two fields appear in the AFM Setup dialog. The `Line frequency` should be set to the frequency at which you want 'fake' EOL trigger signals to come. The `Delay` is the number of pixel time intervals the MLA™ will wait after the 'fake' EOL trigger, before it starts to record a scan line. Setting both of these fields to zero causes no 'fake' EOL signal to be generated, and in this case the AFM Suite relies on the trigger signals recived by MLA™. These two fields should be set to zero when running with external triggers from the host AFM.

5.4 Common AFMs

5.4.1 Nanoscope ™ III, IIIa, IV

ImAFM™ works perfectly well with older Multimode and Dimension AFMs, originally developed by Digital Instruments (DI), then sold by Veeco and currently by Bruker. These are excellent AFM's and there are many good instruments still in service. The upgrade to ImAFM™ is a good way to give these systems a second life and convert them to a modern AFM with the latest surface analysis capabilities. You do not need to upgrade to a NS-V controller and ImAFM™ performance is not compromised with the older NS-III and NS-IV controllers.

Access to the detector signal and cantilever drive are provided via a Signal Access Module (SAM) from Intermodulation Products. This gives you low noise and low cross-talk connection with the host AFM and one switch is all that is required to toggle between normal use of your AFM and ImAFM™. The SAM should be placed in between the microscope and the controller. If there is a Phase Extender Module with the NSIIIa controllers, you should place the SAM in between the microscope and the phase extender module. A general rule of thumb is: Place the SAM directly after the microscope, before the controller or any other ancillary equipment.

You can also use the SAM provided by DI/Veeco/Bruker, which gives access to many other signals, but has some cross-talk between signals which degrades performance, and it requires two switches to change between normal operation and ImAFM™. The actual connections on the DI/Veeco/Bruker SAM depend on which version you have, and which controller you are running. Consult your DI/Veeco/Bruker manual when making these connections. Contact Intermodulation Products if you have any troubles as we have found errors in some versions of these manuals. See *SAM III connections* for an image showing a typical connection.

The trigger signals on the NS-IV controller are accessible from BNC connectors on the back side of the controller. On the NS-III and NS-IIIa controllers they are accessible via BNC connectors located inside the controller box. You need to remove the lid to access these connections. If you would like a more clean solution, you can install connectors on the front of the controller. You need to remove the front panel and drill two holes for this installation. Intermodulation Products can provide the necessary cables and advice.

See *Nanoscope III and IIIa Trigger Access* for images and detailed instructions on accessing the trigger signals.

5.4.2 Nanoscope ™ V

The NS-V controller is currently sold by Bruker for most of their AFMs e.g the Icon and Fast-Scan AFMs. If the NS-V is running a Multimode AFM, or Dimension AFM, the IMP-NS4-SAM gives you signal access. If the NS-V controller is running on a Multimode or Dimension AFM, you can use the same SAM as that for the NSIII and NSIV controller described above. If you are running a Bruker Icon, Fast-Scan, or other more recent AFM using the NSV controller, **Intermodulation Products** has a low noise SAM with access the vertical and latteral detector signals, as well as apply a voltage to the tip. If these connections are sufficient, you do not need the SAM-V module from Bruker. However, if you have the SAM-V, you can use it to make the connections. See *Nanoscope V Connections* for an image showing a typical connection. Contact Intermodulation Products if you would like a dedicated ImAFM™ SAM for the NS-V Icon or Fast-Scan systems.

Access to the trigger signals is on the front panel of the NS-IV controller. See *Nanoscope V Connections* for an image showing the access to the trigger signals.

5.4.2.1 software settings

With the NS-V controller and software v8 and later it is also necessary to configure the software to make the connections required for ImAFM™. In addition to setting the system to **contact mode** and setting the **set-point to 0 Volts**, you need do the following:

To get access to the shaker piezo when the Bruker is set to contact mode, go to the **Microscope** pull-down menu and select **Generic Lockin**. Select the tab for **Lockin-1** and **enable** this lockin. Under **drive routing** select **tapping piezo** and set the **amplitude** to zero. Test the system and if it works, save this workspace as **ImAFM**.

For ImEFM you need to set the software as above, and configure for connection for the tip voltage. Go to the **Advanced Settings** menu (red stop-sign icon). Under the group **other** go to **tip bias control** and select the option **tip bias**. Also in this group go to **sample bias control** and select the option **ground**. Test the system and if it works, save this workspace as **ImEFM**.

5.4.3 Asylum MFP-3D ™

These systems can be found with the older MFP-3D ™ controller, the ARC ™ and ARC2 ™ controller. With all of these controllers, signal access is via BNC connections on the front panel of the controller.

Hardware connections:

- **OUT A** on the MLA™ <-> **In 1** on the controller.
- **OUT 1** on the MLA™ <-> **In 0** on the controller.
- **IN 1** on the MLA™ <-> **Out 0** on the controller.

Signal routing is configured in the software cross-point switch and these settings are most easily made by running a script, o

- **user.alias.l** (left column) <-> **Deflection** (right column).
- **user.alias.d** (left column) <-> **Input.fast** (right column).
- It is necessary to **write** the changes and **Lock** the cross-point
- You can **Read** the changes in the **All** tab and check if **Deflection** is indeed connected to **Input.fast**.
- NOTE: making these changes redefines the alias's for all modes, and therefore Contact mode will not work as normal unless you switch back to the original settings, or restart the Asylum software. Make a note of what **Deflection** is set to before you make these changes manually.

The trigger connections require that you purchase a controller accessory from Asylum Research called the Digital Access Module ™ (DAM). This module connects to a DIN connector on the front of the controller, it has BNC connections for:

- **Port 2 LINE** on the DAM <-> **TRIG 1** on the MLA™.
- **Port 3 FRAME** on the DAM <-> **TRIG 2** on the MLA™.

5.4.4 Asylum Cypher ™

The Cypher AFM runs the ARC or ARC2 controller. The signal connections can be made as with the MFP-3D, but it is preferable if they are made on the so-called 'backpack' of the Cypher itself. The signal connections are:

- **Output 0** on the Cypher backpack <-> **IN 1** on the MLA™.
- **Input 0** on the Cypher backpack <-> **OUT 1** on the MLA™.
- **Input 1** on the Cypher backpack <-> **OUT A** on the MLA™.

The signal routing is made as for the MFP-3D as described above, which is easily done by running the **ImAFM_Cypher.ipf** script.

The trigger connections with the ARC and ARC2 controller are done exactly as with the MFP-3D controller, via the Digital Access Module ™.

5.4.5 JPK Nanowizard ™ III

The connections to the JPK Nanowizard III are shown here: *JPK Nanowizard III Connections*.

As with other systems, ImAFM™ is done with the JPK software in contact mode. Run the IMP-JPK script to open a small GUI which aids in setting all parameters in the JPK software. Press `Setup` to make the JPK system ready for ImAFM™. Note that when doing the noise calibration on a JPK system, it is a good idea to turn off the high voltage to the Z piezo. Press the button `HV-Z OFF` to toggle on and off the high voltage.

5.4.6 NanoTec Electronica

As with other systems, you should set the Nanotec software to run contact mode. In the DA (data acquisition) view, set all other modes to "no" in the Scanning->Scan Options panel.

The Nanotec Dulcinea SPM controller has various inputs and outputs which have to be configured in the Nanotech WSxM Software. Please note that the cantilever drive signal output from the MLA™ requires a special connector at the AFM head or a specific signal routing in the Nanotech Dulcinea controller. Please contact Nanotec for further details.

The signal in port of the MLA™ has to be connected to the BNC port C on the front side of the Dulcinea controller. In the WXsM software, open the Dulcinea BNC monitor and set "C (800 kHz)" to "Normal force" in order to output the cantilever deflection signal.

The feedback signal generated by the MLA™ is put into channel 7 on the rear side of the controller. Open the feedback panel in the WXsM software (View -> Feedback) and set the Feedback Channel to CH 7. Set the the setpoint in the scan control panel to 0 V.

To configure the trigger signals open the User Digital Signals panel (View -> Digital Signals) and choose the configuration "<Customize....>". Set channel 0 to "Image" and channel 3 to "X ramp". Now, the Dulcinea controller outputs an EOF trigger on channel 0 and an EOL trigger on channel 3 at the digital signals connector at the rear side of the controller. Please note, that channel 0 corresponds to pin 2 and channel 3 to pin 9 on the connector.

Before starting to scan, the voltage offset of the feedback channel 7 has to be adjusted. Proceed in the ImAFM™ work flow until you have setup the piezo drive and the feedback my clicking on Setup in the Scanner Panel. Now, open the Input menu in the WxSM software (View -> Inputs), select the tab for channel 7 and click on "Tune offset". If the signal in the scope view of channel 7 is now at 0 V, the tip can be engaged. Make sure that there is no offset filter active when looking at channel 7 in the WXsM software.

5.4.6.1 Hardware Connection images

5.4.6.1.1 SAM III connections

5.4.6.1.2 Nanoscope III and IIIa Trigger Access

5.4.6.1.3 Nanoscope V Connections

5.4.6.1.4 JPK Nanowizard III Connections

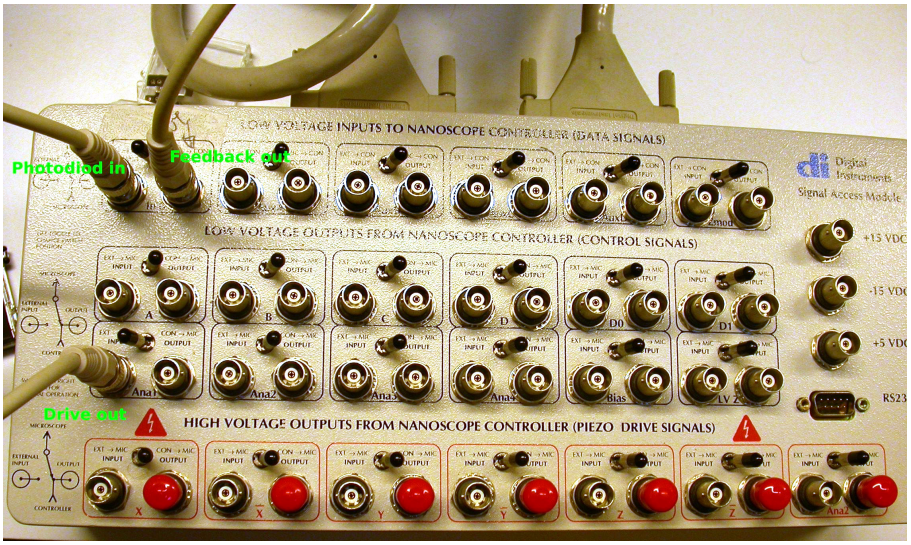


Fig. 1: Typical connections for the SAM from Veeco/Digital Instruments. The images shows the SAM III, connected to a Dimension 3100 AFM. These connections may be different, depending on the controller, AFM, and SAM versions. The switches in the image are in the position for ImAFM™ (left for ImAFM™, right for normal AFM).

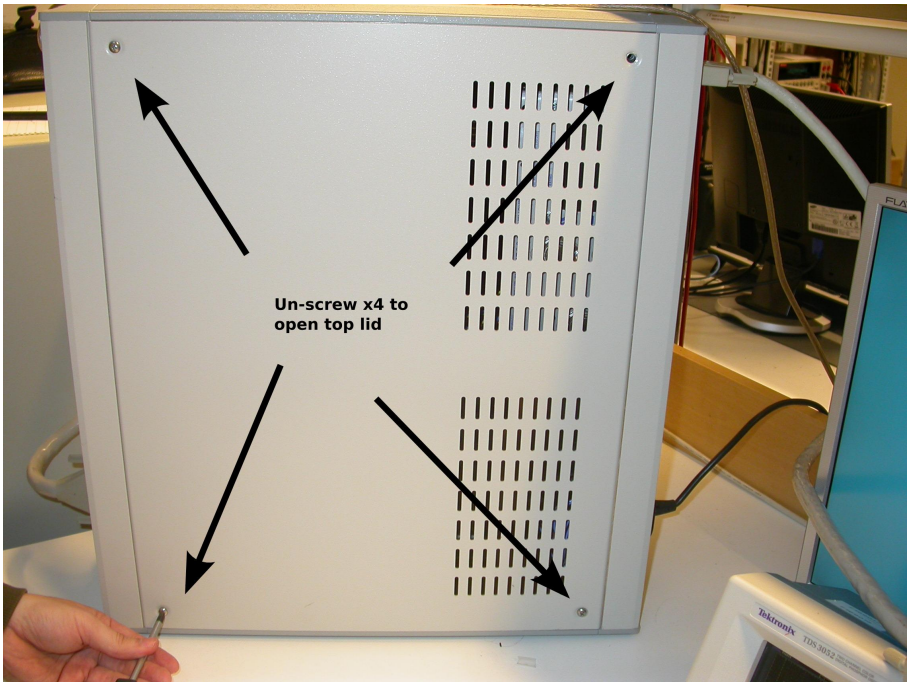


Fig. 2: Remove the 4 screws in each corner of the lid on the NS-III or NS-IIIa controler.

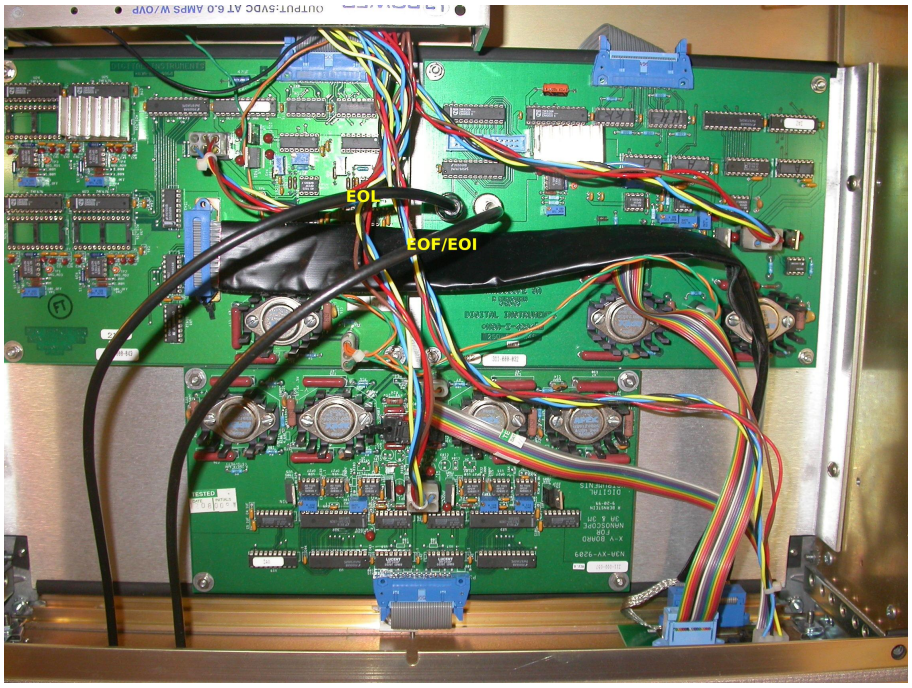


Fig. 3: Connect the End of Line (EOL) and End of Frame (EOF) triggers to the BNC connectors on the circuit board as shown. The images shows cables which bring the these connections to BNC bulk-head connectors mounted on the front panel of the controler.

Fig. 4: BNC bulk-head connectors giving access to EOL and EOF connectors on the front panel.

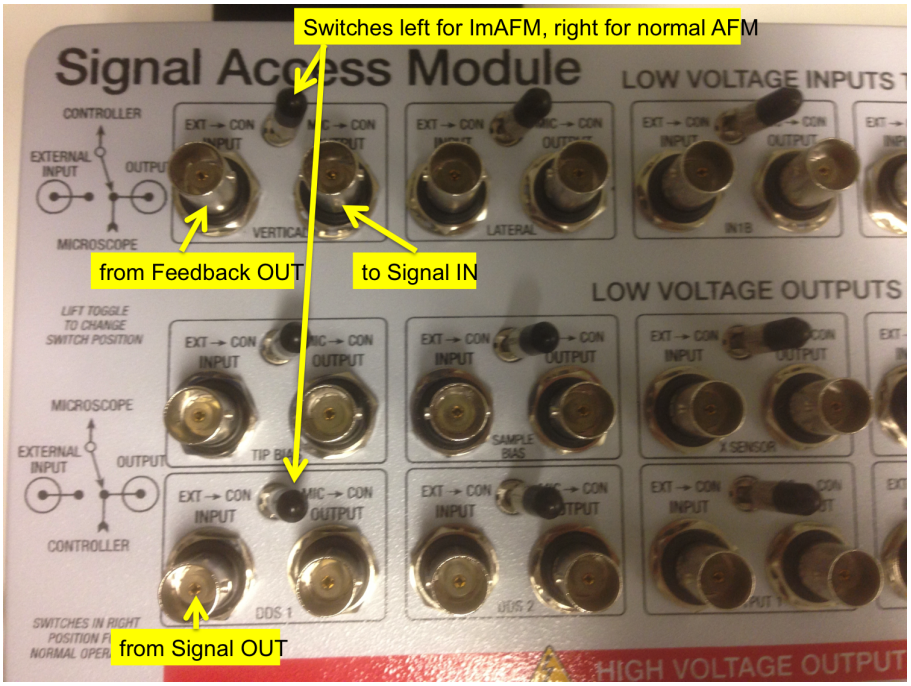


Fig. 5: The connections to the SAM-V from Bruker are identical to the SAM modules from DI and Veeco. However, these SAM modules can not be used interchangably.

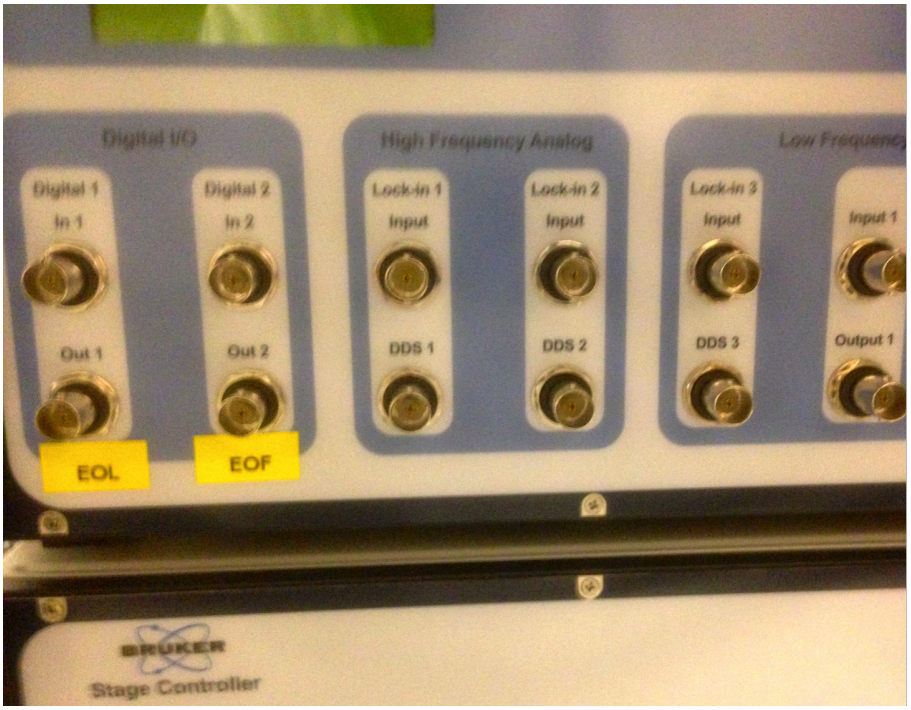
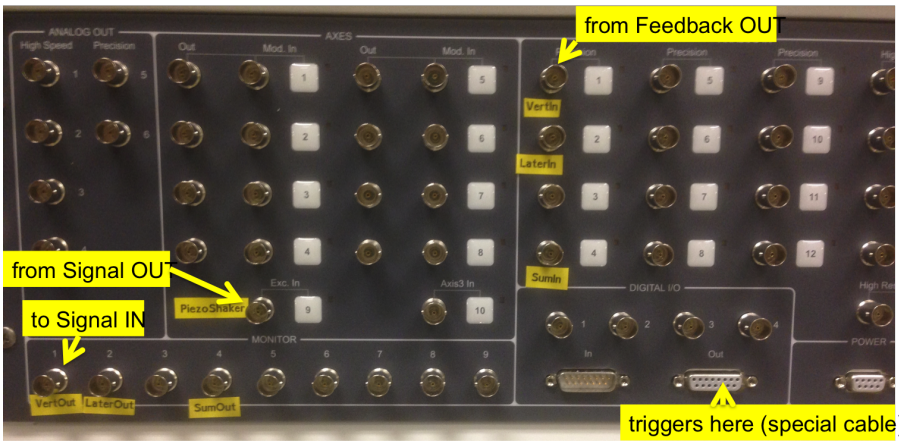


Fig. 6: The EOL and EOF triggers are on the front of the NS-V controller. Make the connections as labeled in the image.


from Feedback OUT
from Signal OUT
to Signal IN
triggers here (special cable)
ANALOG OUT
High Speed
Precision
AXES
Out
Mod. In
VertIn
LaterIn
SumIn
PiezoShaker
Exc. In
Axis3 In
MONITOR
VertOut
LaterOut
SumOut
DIGITAL I/O
In
Out
POWER

CHAPTER

SIX

MULTIFREQUENCY LOCKIN AMPLIFIER

6.1 Version III

For the MLA™-3, please refer to the online manual at https://intermodulation-products.com/manuals/IMP-MLA_user_manual/mla.html

6.2 Version II

For the MLA™-2, please refer to the online manual at https://intermodulation-products.com/manuals/IMP-MLA_user_manual/mla_2.html

6.3 Version I

Version I of the MLA™ (or ImLA™) has fewer connections and can handle fewer frequencies.

Front-side connections

- `Signal OUT` sends the drive frequency comb to the shaker piezo of the AFM. The system is configured to work nicely with standard AFMs. Internal jumpers can be moved to change the output voltage range. Contact Intermodulation Products if you want different output range.
- `Feedback OUT` delivers a error signal to the host AFM, where zero volts corresponds to the set-point in the AFM Suite. When the host AFM is set to contact mode, with set-point zero in the host AFM software, The voltage sent by the Feedback Out port constitutes the error signal in the control of the AFM feedback. The feedback gains are set on the host AFM.
- `Signal IN` receives the detector signal from the AFM. The system is configured to work nicely with standard AFMs. Attenuators can be used to reduce the amplitude of the input signal if necessary. Contact Intermodulation Products if you want a different input range.
- `EOF TRIG IN` is the **End Of Line** signal which is generated by the host AFM and received by the MLA™.
- `EOL TRIG IN` is the **End of Frame** signal which is generated by the host AFM and received by the MLA™.

Back-side connections

- `+5V DC` powers the MLA™. One should always use the power supply delivered with MLA™. Using any other supply voids the warranty.
- `Ethernet` The MLA™ communicates with the computer via the Ethernet connection. The MLA™ may be directly connected to the Ethernet connection on the computer, or it may be be put behind an Ethernet router, switch or gateway.
- `10 MHz OUT` is a TTL sync signal for locking the MLA™ clock to external equipment.

6.3.1 Version I Firmware

Two basic firmware versions are available

- IMP-2-23 : Two drive frequencies at the `Signal Out` port. 32 frequencies analyzed at the `Signal IN` port.
- IMP-24-24 : 24 drive frequencies at the `Signal Out` port and the same 24 frequencies are analyzed at the `Signal IN` port.

CHAPTER

SEVEN

TROUBLE SHOOTING

Below is a list of problems that may arise, and solutions. Contact Intermodulation Products if you are having problems getting your system to operate <info@intermodulation-products.com>. There is a hidden program long file in each session folder **.internallog.dat** which the company representative may need to see if the problem can not be solved over the phone.

Problem: The software crashed but the process is still running in the computer and I can not stop it.
Solution: In Windows: start the task manager (control+shift+escape). Look for **imp_suite.exe** and kill that process.

Problem: I get only a very weak signal and I do not see any resonance when I do a frequency sweep.
Solution: Did you make sure that the hardware switch on the signal access module in is the correct position?

Problem: I can not get a good fit to the noise calibration data.

Solution:

- There may be a spurious signal coming from your AFM which happens to be close to resonance. Change cantilever and avoid the frequency of the spurious signal.
- You can exclude data points to avoid frequency bands where spurious signals and additional noise is getting in to you measurement.
- There may be an extra (not thermal) source of noise which is driving your cantilever. Make sure the drive from the host AFM is off and disconnected. Try turning off the high voltage drive to the Z-piezo of a tip-scanning AFM.

Problem: When I open the software, only get the Analysis and Session Overview icons in the work flow.
Solution: This means that the software can not find the MLA™ on the ether net. Check that the MLA™ is connected to the ether net, and verify that the computer has ether net connection to the MLA™.

CHAPTER

EIGHT

CHANGELOG

This list oulines major new features and bug fixes between releases of the ImAFM™ suite. Each release typically also includes additional minor bugfixes and smaller new features.

Version 4.0.2

Release date: 2019-11-07

Note: see changenote for 4.0.1 regarding updating from earlier versions.

Improvements

- ImEFM turns on output amplifiers if needed.

Bugfixes

- Fixed exporting force curves from data tree.
- Fixed 3D-viewer on Windows.
- Fix to ImFFM which could freeze.

Version 4.0.1

Release date: 2019-09-09

Due to changes in the hardware bootup sequence the following commands must be run from the built-in Python shell once for every MLA (if it was shipped with an older version).

```
from mlaapi import setup
setup.upload_bootware(mla.hardware.ip_address)
```

Additionally, if the IP-address has been changed from the default 192.168.42.50 you must also run

```
setup.set_ip_address(mla.hardware.ip_address, mla.hardware.ip_address)
```

Improvements

- Various improvements to Frequency sweep: voltage kept at zero after sweep, individual curves can be shown / hidden from toolbar.

Bugfixes

- Removed transients appearing during startup.
- Fix bug in ADFS force volume export.
- Various bugfixes in cantilever calibrator: allow <1kHz range, allow user to re-zoom while acquiring data.

Version 4.0

Release date: 2019-07-24

We have made a major transition of the software under the hood. It is now based on Python 3 instead of Python 2, and is therefor more future proof. With this upgrade we are simulatenously dropping support for MLA gen2, there will be no parallell releases of gen2-software. Data obtained with MLA gen2 hardware can however still be opened and analysed in this and future versions to make use of the latest analysis techniques.

New features

- Support for MLA gen3.1 (serial numbers >= 3100).
- Stream Recorder saves DC values of all four inputs with lockin data.
- Verbose flag added (-v) to show more warnings and error messages

Improvements

- Model fit parameter maps speed up.
- New entry-point to software with more obvious name "start_afm.py"

Bugfixes

- Fix in Stream Recorder with low df.

Version 2.8 / 3.2

Release date: 2019-02-22

New features

- "Moving surface model" analysis plugin included by default.
- Linear Discriminant Analysis (LDA) included in BlackboxAnalysis plugin

Improvements

- DMT_EXP model renamed DMT_ExpDamp to clarify exponential damping.

Bugfix

- Fixed set new calibration from Session overview.
- Several fixes related to non-square scans.
- Removed a lot of unecessary warnings when booting up (start software with –verbose to see more warnings).

Version 2.7.1 / 3.1.1

Release date: 2018-09-05

Improvements

- Frame title shows if we are in OFFLINE mode.

Bugfixes

- Fix name collision (version, version.dll) which prevented software from starting on Window 10.
- Fix some deprication warnings.
- Fix a connection time-out error that effected some machines.

Version 2.7 / 3.1

Release date: 2018-07-05

New features

- Compatibility with MLA generation 3 (version 3.1)
- GUI to change config.ini settings (Advanced->All settings)
- New menu "MLA" for to access additional lockin functionality
- Fitted force model is now displayed also in the Fi and Fq-plots.
- K-means clustering: ability to cluster either in force or deflection domain, new gui to select file.

Improvements

- GUI library updated to wxPython 4

- Message log does not pop up on every error, instead a message is printed in the status bar in the bottom of the screen. Double-click to see whole message.
- Addonviewer: colorscale selector (right-click on colorbar) and scroll zoom implemented.

Bugfixes

- PNG export from model fit did sometimes not work if no scan size had been set.
- New firmware for gen 3 (205), fixes a small glitch in feedback output signal.

Version 2.6

Release date: 2018-03-27

New features

- Lockin monitor to easily view lockin values in Manual Setup panel.
- New Script Panel for scripts in the style of the IMP MLA software.
- Previous Scripts have been renamed into Plugins. NOTE: Old scripts have to be manually moved from IMP Sessions and settingssettingsscripts to settingsafm_plugins.
- Input and output port selector in Frequency Sweep.
- Ability to set a different DC offset for each lockin time window.
- ScanData addon images can be viewed directly in the Session overview.

Improvements

- A lot of stability fixes in the underlying MLA library.
- Significant speed-up of noise calibration.
- Easier to integrate customized analysis in Session overview.
- ImEFM: allow changing DC offset while scanning.

Bugfixes

- Fixed a bugs which could prevent processing of retrace in AFMs with only one trigger per line, such as Asylum Cypher.
- mla.osc.ad_is_in_safe_range was not properly detecting a clipped input.
- Fix resolution of non-square images in Gwyddion loader.
- Frequency Sweep, fixed limits for max V and frequency.

Version 2.5.2

Release date: 2017-09-04

New features

- Principal-component-analysis (PCA) added to black box model scripts.
- “Smooth pixel tool”, calculate force curves from the avarage of several pixels.
- Physical ports for vertical and horizontal signal can be set from config.ini.
- New panel (Menu->Advanced->Message Log) to see error messages and debug print outs.

Improvements

- Improved interface for measure just lifted.
- Improved color scales with more linear lightness.

Bugfixes

- reset_tones_on_new_pixel was set automatically when running start_lockin, affecting some custom imaging modes.

- Software did not start after having reached 9999 measurement files.

Version 2.5.1

Release date: 2017-04-04

Bugfixes

- Ability to import height data from Nanoscope version 9.2
- Corrected scaling of Nanoscope height files when height sensor calibration existed, but height sensor was not used.
- Tweaks to make measure just lifted more stable.
- Modified behaviour of session log book to reduce the risk of deleting log entry by mistake.

Version 2.5

Release date: 2017-03-02

New features

- Background force compensation using new measure lift function.
- Changes to force curve plotting: ADFS default (no longer polynomial), plot only modulated range.
- Stavitzky-Golay smoothing of ADFS force curves.
- Speed-up of FI and FQ calculation.
- Speed-up of ADFS force curves, force volume export almost 40x faster.
- New Mode: Intermodulation Frictional Force Microscopy (ImFFM) with torsional eigenmode calibration.
- Support for input multiplexing feature of the MLA (needed for ImFFM) available from Drive Constructor.
- Mouse wheel zoom features in all plots and images.
- Drag-and-drop support for opening files and folder.
- Graphs showing convergence of calibration.
- Simplified ImAFM set-up (automatic modulation range).
- Script for calculating True Topography using feature on FI FQ curves.
- Level data tool in analysis pannel
- New scheme in ImEFM for improved surface tracking.
- Improvements to Drive Constructor.
- Improvements to 3D viewer, use amplitude and phase images as texture.
- Update and improvement of the User Manual.

Bugfixes

- Lasso tool picks data from correct scan pass (trace / retrace).
- Zoom fixed in ImEFM.
- Improvements and fixes to Gwyddion loader
- Can now load newer JPK files

Installation notes

- Application has been renamed from “IMP Software Suite” to “IMP AFM Software Suite”. Manual uninstallation of previous version required to clean up start-menu and desktop icon.

Version 2.4.1

Release date: 2016-10-04

New features

- Frequency sweep: reverse sweep direction.
- Manual setup panel for setting up the lockin.
- New ScanData file format version (9) keeping track of scan type (ImAFM, ImEFM etc).
- Model fit parameter maps additionally stored as ASCII files.
- Lifted motion displayed together with free motion in show meta data browser.

Bugfixes

- ScanData: fixed export to ascii and sampling frequency was not correct when using drive constructor.
- Easier to use binary Gwyddion loader for ".imp"-files. Does not require any special version of or python Gwyddion for windows. Note: requires installation of Visual Studio 2015 redistributables from Microsoft.

Version 2.4

Release date: 2016-06-30

New features

- New MLA firmware: MLA can measure DC-component at all inputs simultaneously.
- Script to record ramps / approach curves using DC signal.
- Support fo NT-MDT triggers.
- New format for storing calibration in ScanData files.

Bugfixes

- Improved interpretation of EOL and EOF signals from AFM. Trace/retrace and left/right flipping is automatically syncronized after EOF trigger. First and last scan line of images now stored correctly.
- Fix model fit force curves not appearing when using area tool.

Version 2.3

Release date: 2016-04-06

New features

- Measure lift.
- When saving an image directly from the GUI you will now get a dialog box where you can type the resolution. There is also a new option in the "Save as type" field named "Native resolution png". With this option pure png files will be saved with the same resolution as the data. In addition, separate files with the colorbars will be generated. This could be practical if you want to assemble the figure yourself in for example powerpoint.
- There is a hold-off time (default 1 ms) before a new trigger can be accepted to avoid bouncing.
- Phase feedback using port OUT C. This is activated from the python shell with:

  mla.feedback.activate_phase()

  The full phase range of -pi to +pi corresponds to a voltage range of -1 V to +1 V. You can offset the voltage with mla.feedback.set_phase_offset(val), when val is -32768 to 32767.

  A more convenient way to adjust the offset may be to use mla.feedback.set_current_phase_to_value(V), where V is a voltage (-4 to +4). This will read the current phase value and adjust the offset of the phase feedback so that the resulting voltage on port OUT C is the specified value.

  Before you can do anything with the phase feedback, the normal feedback must be set up as usual with the "Setup" button in the "Scanner" or the "Advanced setup" perspectives. This phase feedback is to be considered experimental.
- Gwyddion plug-in to open ScanData files. Instructions for how to install it in the folder C:Program FilesIMP Software SuiteExtToolsGwyddion

- New icon in the start menu to start the software in Debug mode.

Bugfixes

- Better handling of unusual values of amplitude and df.
- The rotation of merged Asylum images is fixed.
- The data transfer speed is improved so it should be possible to scan faster.

Version 2.2

New features

- Plot of free oscillation accessible from data tree
- Many additional new features

Bugfixes

- Live parameter map works with half scan speed
- Additional minor tweaks including description of units in imp-files

Version 2.1

New features

- Improved smaller icon set (white emblem on blue background)
- Many additional new features and bugfixes

Version 2.0

New features

- First version software to use the second generation multifrequency lockin amplifier (MLA)

CHAPTER

NINE

REFERENCES

BIBLIOGRAPHY

[Borgani-2014] Intermodulation electrostatic force microscopy for imaging surface photo-voltage Riccardo Borgani, Daniel Forchheimer, Jonas Bergqvist, Per-Anders Thorén, Olle Inganas, and David B. Haviland Applied Physics Letters 105, 143113 (2014); doi: 10.1063/1.4897966 Borgani_2012

[Borgani-2017] Background force compensation in dynamic atomic force microscopy. Riccardo Borgani, Per-Anders Thorén, Daniel Forchheimer, Illia Dobryden, Si Mohamed Sah, Per Martin Claesson, David B. Haviland Borgani_2017

[Forchheimer-2012] Model-based extraction of material properties in multifrequency atomic force microscopy. Daniel Forchheimer, Daniel Platz, Erik A. Tholén, and David B. Haviland. Phys. Rev. B. **85**, 195449 (2012). Forchheimer_2012

[Forchheimer-2013] Simultaneous imaging of surface and magnetic forces. Daniel Forchheimer, Daniel Platz, Erik A. Tholén and David B. Haviland. Appl. Phys. Lett. **103**, 013114 (2013) Forchheier_2013

[Green-2002] Torsional frequency response of cantilever beams immersed in viscous fluids with applications to the atomic force microscope. Christopher P. Green and John E. Sader Jour. Appl. Phys. **92**, 6262 (2002).

[Haviland-2016] Quantitative force microscopy from a dynamic point of view. David B. Haviland Current Opinion in Colloid & Interface Science **27** 74–81 (2016). Haviland_2016

[Higgins-2006] Noninvasive determination of optical lever sensitivity in atomic force microscopy. M. J. Higgins, R. Proksch, J. E. Sader, M. Polcik, S. Mc Endoo, J. P. Cleveland and S. P. Jarvis. Rev. Sci. Instr. **77**, 013701 (2006).

[Hutter-2010] Reconstructing Nonlinearities with Intermodulation Spectroscopy. C. Hutter, D. Platz, E. A. Tholén , T. H. Hansson and D. B. Haviland. Phys. Rev. Lett. **104**, 050801 (2010). Hutter_2010

[Platz-2008] Intermodulation atomic force microscopy. D. Platz, E. A. Tholén, D. Pessen and D. B. Haviland. Appl. Phys. Lett. **92**, 153106 (2008). Platz_2008

[Platz-2012a] The role of nonlinear dynamics in quantitative atomic force microscopy. D. Platz, D. Forchheimer, E. A. Tholén and D. B. Haviland. Nanotechnology **23**, 265705 (2012). Platz_2012a

[Platz-2012b] Interaction imaging with amplitude-dependence force spectroscopy. Daniel Platz, Daniel Forchheimer, Erik A. Tholén and David B. Haviland. Nature Commu. **4**, 1360 (2012). doi:10.1038/ncoms2365 Platz_2012b

[Platz-2013a] Interpreting force and motion for narrow-band intermodulation atomic force microscopy. Daniel Platz, Daniel Forchheimer, Erik A. Tholén and David B. Haviland. Beilstein J. Nanotechnol. **4**, 45 (2013). Platz_2013a

[Platz-2013b] Polynomial force approximations and multifrequency atomic force microscopy. Daniel Platz, Daniel Forchheimer, Erik A. Tholén and David B. Haviland. Beilstein J. Nanotechnol. **4**, 352 (2013). Platz_2013b

[Sader-1998] Frequency response of cantilever beams immersed in viscous fluids with applications to the atomic force microscope. J. E. Sader. Jour. Appl. Phys. **84**, 64 (1998).

[Sader-2005] General scaling law for stiffness measurement of small bodies with applications to the atomic force microscope. J. E. Sader *et al.* Jour. Appl. Phys. **97**, 124903 (2005).

[Sader-2012] Spring constant calibration of atomic force microscope cantilevers of arbitrary shape. J. E. Sader *et al.* Rev. Sci Instr. **83** 103705 (2012).

[Tholen-2011] The intermodulation lockin analyzer. E. A. Tholen, D. Platz, D. Forchheimer, V. Schuler, M. O. Tholén, C. Hutter and D. B. Haviland. Rev. Sci. Instr. **82**, 026109 (2011). Tholen_2011

[Thoren-2017] Calibrating torsional eigenmodes of micro cantilevers for dynamic measurement of frictional forces. P.-A. Thorén et al. to be published 2017

INDEX

A

accuracy, 32
Acquire noise data, 8
ADFS, 21, 22
ADU, 8
Amplitude Dependent Force Spectroscopy, 22
Amplitude image, 12
Analysis, 15
Area inspector, 14

B

Back tool, 13
back-side connections, 65
Background force compensation, 23
Batch parameter maps, 16

C

Calibration, 6
Calibration method, 7
Calibration Result, 8
Cantilever, 7
Cantilever calibration, 6
Color Bar, 12
comments, 17
connections, 65
Contact potential difference, 25
current calibration, 9
Cz, 27

D

Data Acquisition, 8
Detection noise floor, 8
Detector noise floor, 32
detector responsivity, 8
detector sensitivity, 8
dissipated energy, 20

E

Electrostatic Force Microscopy, 25
Exclude data, 8

F

Feedback, 30
FI FQ, 19
file association, 15
finding the resonance, 6
Fitting noise data, 8
Flip right-left, 12
Flip up-dpwn, 12
flipping images, 16
Fluid, 7
Force, 19
Force curves, 19
Force Inspector, 19
Force Inspector settings, 23
Force Reconstruction, 21
force sensitivity, 8
Forward tool, 13
Frequency Mode, 30
Frequency mode, 30
Frequency Sweep, 6
front-side connections, 65

G

Gaussian filter, 15
get_amplitudes() (*afmapp.Scanner.ScanData.ScanData method*), 45
get_calib_mode_index() (*afmapp.Scanner.ScanData.ScanData method*), 45
get_cantilever_noise() (*afmapp.Scanner.ScanData.ScanData method*), 45
get_data() (*afmapp.Scanner.ScanData.ScanData method*), 46
get_detector_noise() (*afmapp.Scanner.ScanData.ScanData method*), 46
get_df() (*afmapp.Scanner.ScanData.ScanData method*), 47
get_drive_amplitudes() (*afmapp.Scanner.ScanData.ScanData method*), 47
get_drive_karray() (*afmapp.Scanner.ScanData.ScanData method*), 47
get_drive_phases() (*afmapp.Scanner.ScanData.ScanData method*), 48
get_endtime() (*afmapp.Scanner.ScanData.ScanData*

method), 48
get_external_filename() (afmapp.Scanner.ScanData.ScanData method), 48
get_f0() (afmapp.Scanner.ScanData.ScanData method), 48
get_feedback() (afmapp.Scanner.ScanData.ScanData method), 48
get_free() (afmapp.Scanner.ScanData.ScanData method), 48
get_frequencies() (afmapp.Scanner.ScanData.ScanData method), 49
get_fs() (afmapp.Scanner.ScanData.ScanData method), 49
get_imp_str_list() (afmapp.Scanner.ScanData.ScanData method), 49
get_input_ports() (afmapp.Scanner.ScanData.ScanData method), 49
get_invOLR() (afmapp.Scanner.ScanData.ScanData method), 49
get_k() (afmapp.Scanner.ScanData.ScanData method), 50
get_karray() (afmapp.Scanner.ScanData.ScanData method), 50
get_lift() (afmapp.Scanner.ScanData.ScanData method), 50
get_loaded_filename() (afmapp.Scanner.ScanData.ScanData method), 50
get_metadata_ascii() (afmapp.Scanner.ScanData.ScanData method), 51
get_mla_calibration() (afmapp.Scanner.ScanData.ScanData method), 51
get_mla_firmware_version() (afmapp.Scanner.ScanData.ScanData method), 51
get_original_filename() (afmapp.Scanner.ScanData.ScanData method), 51
get_output_mask() (afmapp.Scanner.ScanData.ScanData method), 51
get_phases() (afmapp.Scanner.ScanData.ScanData method), 51
get_Q() (afmapp.Scanner.ScanData.ScanData method), 45
get_resolution() (afmapp.Scanner.ScanData.ScanData method), 51
get_samples_per_pixel() (afmapp.Scanner.ScanData.ScanData method), 51
get_scan_rate() (afmapp.Scanner.ScanData.ScanData method), 51
get_scan_size_x() (afmapp.Scanner.ScanData.ScanData method), 52
get_scan_size_y() (afmapp.Scanner.ScanData.ScanData method), 52
get_scan_subtype() (afmapp.Scanner.ScanData.ScanData method), 52
get_scan_type() (afmapp.Scanner.ScanData.ScanData method), 52
get_setpoint() (afmapp.Scanner.ScanData.ScanData method), 52
get_starttime() (afmapp.Scanner.ScanData.ScanData method), 52
get_tip_Vac() (afmapp.Scanner.ScanData.ScanData method), 52
get_tip_Vdc() (afmapp.Scanner.ScanData.ScanData method), 52
given_list_is_empty() (afmapp.Scanner.ScanData.ScanData method), 52

H

has_calibration() (afmapp.Scanner.ScanData.ScanData method), 53
Home tool, 13
host AFM files, 15
Hydrodynamic function, 7
hysteresis in FI FQ curves., 19

I

Image Settings, 12
image toolbar, 13
ImEFM, 25
ImLA, 29, 65
IMP Control, 12
IMP Session, 4
Import host AFM data, 15
independent lockins, 65
inputs, 65
Intermodulation Lockin Analyzer, 29, 65
intermodulation measurement, 29
inversion, 18
invOLS, 8

L

Line direction, 12
Line Inspector, 13
load_from_file() (afmapp.Scanner.ScanData.ScanData method), 53
Logbook, 17

M

Measure lift, 11

minimum detectable force, 8, 32
MLA, 29, 65
Mode stiffness, 8
Model Fit, 22
Mouse wheel zooming, 13
multifrequency lockin amplifier, 29, 65

N

Noise calibration, 30
noise floor, 8
Noise measurement, 8
noninvasive calibration, 30

O

optical lever sensitivity, 8
oscillation range, 11
oscillator approximation, 30
outputs, 65

P

Pan-Zoom tool, 13
Parameter maps, 24
Phase image, 12
PID control, 30
Pixel Inspector, 13
plan-view dimensions, 7
Polynomial reconstruction, 21

Q

quadrature forces, 19
Quality factor, 8
Quantitative Analysis, 18

R

Reconstruction Methods, 21
Reference calibration, 7
reset_lift() (*afmapp.Scanner.ScanData.ScanData method*), 53
Resonance frequency, 8
responsivity, 8, 32
Retrace, 12
Run calibration, 8

S

Sader constants, 7
Save image tool, 13
save_to_ascii() (*afmapp.Scanner.ScanData.ScanData method*), 53
save_to_file() (*afmapp.Scanner.ScanData.ScanData method*), 53
saving images, 13
Scan rate, 12
ScanData (*class in afmapp.Scanner.ScanData*), 44
scanner panel, 10
Scanning, 10
Select range, 8
sensitivity, 32
Session Logbook, 17
Session Overview, 15
Set-point, 11
set_amplitudes() (*afmapp.Scanner.ScanData.ScanData method*), 53
set_calibration() (*afmapp.Scanner.ScanData.ScanData method*), 53
set_cantilever_noise() (*afmapp.Scanner.ScanData.ScanData method*), 54
set_detector_noise() (*afmapp.Scanner.ScanData.ScanData method*), 54
set_df() (*afmapp.Scanner.ScanData.ScanData method*), 54
set_endtime() (*afmapp.Scanner.ScanData.ScanData method*), 54
set_external_filename() (*afmapp.Scanner.ScanData.ScanData method*), 54
set_f0() (*afmapp.Scanner.ScanData.ScanData method*), 54
set_feedback() (*afmapp.Scanner.ScanData.ScanData method*), 55
set_free() (*afmapp.Scanner.ScanData.ScanData method*), 55
set_frequencies() (*afmapp.Scanner.ScanData.ScanData method*), 55
set_fs() (*afmapp.Scanner.ScanData.ScanData method*), 55
set_input_ports() (*afmapp.Scanner.ScanData.ScanData method*), 55
set_invOLR() (*afmapp.Scanner.ScanData.ScanData method*), 56
set_k() (*afmapp.Scanner.ScanData.ScanData method*), 56
set_lift() (*afmapp.Scanner.ScanData.ScanData method*), 56
set_mla_all_settings() (*afmapp.Scanner.ScanData.ScanData method*), 56
set_mla_calibration() (*afmapp.Scanner.ScanData.ScanData method*), 57
set_mla_firmware_version() (*afmapp.Scanner.ScanData.ScanData method*), 57
set_output_mask() (*afmapp.Scanner.ScanData.ScanData method*), 57
set_phases() (*afmapp.Scanner.ScanData.ScanData method*), 57
set_Q() (*afmapp.Scanner.ScanData.ScanData method*), 53
set_resolution()

(*afmapp.Scanner.ScanData.ScanData method*), 57
set_samples_per_pixel() (*afmapp.Scanner.ScanData.ScanData method*), 57
set_scan_rate() (*afmapp.Scanner.ScanData.ScanData method*), 57
set_scan_size_x() (*afmapp.Scanner.ScanData.ScanData method*), 57
set_scan_size_y() (*afmapp.Scanner.ScanData.ScanData method*), 57
set_scan_subtype() (*afmapp.Scanner.ScanData.ScanData method*), 57
set_scan_type() (*afmapp.Scanner.ScanData.ScanData method*), 57
set_setpoint() (*afmapp.Scanner.ScanData.ScanData method*), 58
set_starttime() (*afmapp.Scanner.ScanData.ScanData method*), 58
set_tip_Vac() (*afmapp.Scanner.ScanData.ScanData method*), 58
set_tip_Vdc() (*afmapp.Scanner.ScanData.ScanData method*), 58
Setup, 10
Signal Inspector panel, 19
Simple Harmonic Oscillator Fit, 8
Smooth image, 15
Smoothing dialog, 15
Spectrum fit, 21
spring constant, 8
Status Bar, 14
subplots, 13
Swap, 12
swapping trace/retrace, 16

T

Temperature, 7
Thermal noise force, 8, 32
three dimensional parameter maps, 17
three dimensional viewer, 17
Time mode, 30
tip-surface forces, 18
tip-surface voltage, 25
torsional calibration, 10
Trace, 12
triggering, 65

U

Use shading, 19
user comments, 17

V

Vcpd, 27

W

Width of resonance, 8
work, 20
Work Flow, 4, 5

Z

Zoom tool, 13